Ministry of Higher Education and Scientific Research
وزارة التعليم العالي والبحث العلمي

Badji Mokhtar Annaba University
Université Badji Mokhtar – Annaba
Faculty of Technology

IT Department

جامعة باجي مختار – عنابـــة
كـليـة التكنـولـوجيـا
قسـم الاعلام الالـي

Thesis

Presented to obtain the diploma of

Doctorat En-Sciences

Specialty: Informatique

By:
HALLACI Samir

Theme :

Detection and Understanding of Objects in an Image Sequence:
Application to Medical Imaging

Thesis defended ….. 02/06/2024 ……. before the jury composed of :

N°	Last name and first name	Rank	Establishment	Quality
01	SOUISSI-MESLATI Labiba	Prof.	Université Badji Mokhtar -Annaba	Chair
02	SERIDI Hamid	Prof.	Université 08 Mai 1945 -Guelma	Supervisor
04	FARAH Nadir	Prof.	Université Badji Mokhtar -Annaba	Examiner
05	BENMOUHAMMED Mohamed	Prof.	Université Constantine 2	Examiner
06	MAZOUZI Smaine	Prof.	Université 20 aout 1955 - Skikda	Examiner

Dedication

To
My Parents, &
My Wife,
To
My Kids. Abderrahmen, Anes and Ouais.
To
My Brother and Sisters and their kids,&
My family,
for their sacrifices, patience, love, and trust in me. They have done everything for my happiness and success. No dedication can express what I owe them; May God bless them with good health and a long life.

.
.
.

Special thanks To
My Uncle Ali, for believing in me.
And all my Uncles : Mourad, Said, Hacen, Boualem, Lounes, Salim, Youcef, Ami Kader.
To a special friend: Nacer Abderrahim.

Acknowledgement

Thank you, Allah, for providing me the capacity to think and write, the courage to hold into my convictions, and the perseverance to see my dreams through to completion.

In the journey of knowledge, gratitude becomes the compass guiding us through the labyrinth of learning. I am humbled and deeply grateful to those who have illuminated my path with their wisdom, patience, and support.

To my esteemed Supervisor *Pr. Seridi Hamid*, your unwavering guidance and profound insights have been the beacon that steered me through the complexities of this long academic voyage. Your mentorship has not only enriched my research but has also shaped me into a more diligent and discerning scholar. I am indebted to your wisdom, dedication and support.

To my beloved family, your boundless love and encouragement have been my steadfast anchor amidst the tempests of academia. Your sacrifices and unwavering belief in my potential have been the driving force behind my perseverance. Your resilience and unwavering support have been the bedrock upon which I stand today.

To my special colleagues, friends and brothers, *Pr. Farou Brahim* and *Pr. Kouahla Zineddine*, for everything, especially for believing in me.

To the members of the jury, who will soon undertake the task of reviewing and evaluating this thesis, I extend my sincere appreciation. Your expertise and thoughtful consideration will undoubtedly contribute to the refinement and improvement of this work. I am grateful for your time, for your commitment to scholarly excellence and, and valuable feedback.

To my esteemed colleagues and friends, in LabSTIC and the Computer Science department, your camaraderie and intellectual exchange have been a source of inspiration and motivation. I am privileged to have shared this journey with such esteemed companions.

To the countless souls whose wisdom has graced the pages of literature and research, your contributions have laid the foundation upon which this thesis stands. I am indebted to the scholars, past and present, whose intellectual legacy continues to shape the discourse of our fields.

Special Thanks go to Madame Seridi and Madame Gabriela Kouahla, for being an invisible support for us.

In the boundless expanse of knowledge, we are but humble seekers traversing the shores of understanding. May our pursuits be guided by humility, curiosity, and reverence for the mysteries that lie beyond our grasp.

With heartfelt gratitude, **HALLACI SAMIR**

" اكتشاف وفهم الأجسام في تسلسل الصور : التطبيق على التصوير الطبي"

الملخص:

تستكشف هذه الدراسة استخدام الكشف عن الكائنات وفهمها في سلسلة من الصور لتشخيص كوفيد-19، إلى جانب تحليل صور الأشعة السينية الفردية للصدر. من خلال استخدام الشبكات العصبية التلافيفية (CNNs) مع خطوات معالجة مسبقة مناسبة، يستخرج النموذج الأنماط والتبعيات الزمنية من بيانات التصوير المتسلسلة، مما يساعد في مراقبة تطور المرض وفعالية العلاج والتنبؤ بالنتائج. لقد انتشرت جائحة كوفيد، التي نشأت في ووهان بالصين في عام 2019، في أكثر من 200 دولة، مما فرض عبئًا كبيرًا على الأنظمة الصحية العالمية والاقتصاد العالمي. وقد أصبحت الحاجة الملحة إلى طرق تشخيص أسرع وأكثر سهولة في الوصول إليها، خاصة في المناطق ذات الموارد المحدودة. ويمكن أن يساعد الكشف المبكر في خفض معدلات الوفيات ومنع انتقال المرض. وعلى الرغم من استخدام اختبارات تفاعل البوليميراز المتسلسل بالنسخ العكسي (RT-PCR) على نطاق واسع كطريقة موثوقة للكشف عن الفيروس، إلا أن لها عيوبًا مثل كونها تستغرق وقتًا طويلاً ومكلفة ومحفوفة بالمخاطر بالنسبة للعاملين في مجال الرعاية الصحية بسبب المخالطة اللصيقة للمرضى.

واستجابةً لهذه التحديات، تم استخدام تقنيات التصوير الطبي مثل الأشعة السينية لفحص كوفيد-19 نظراً لسرعتها وأمانها وتوافرها على نطاق أوسع مقارنةً بالطرق التقليدية. وقد أظهر دمج الذكاء الاصطناعي AI في معالجة الصور نتائج واعدة في التمييز بدقة بين الصور الشعاعية للصدر الطبيعية والمريضة، مما يساعد في التشخيص المبكر وإدارة المريض. في هذه الدراسة، نقترح نموذجًا للتعلم العميق لتحليل صور الأشعة السينية للصدر لفيروس كورونا المستجد (كوفيد-19)، بهدف معالجة قيود اختبار تفاعل البوليميراز المتسلسل وتعزيز كفاءة التشخيص. تشمل التحديات محدودية بيانات التدريب، مما يؤثر على تعميم النموذج. تتضمن هذه الطريقة المعالجة المسبقة لتوازن الألوان البسيط، وتجزئة الرئة باستخدام شبكة U-NET، وزيادة البيانات لاختلال توازن الفئات، وشبكة CNN خفيفة الوزن من أجل كفاءة الحساب. تتألف الطريقة المقترحة من خطوة معالجة مسبقة لتحسين جودة الصورة باستخدام أبسط توازن الألوان، يليها تجزئة الرئة باستخدام شبكة U-Net لإزالة البيانات غير ذات الصلة وتعزيز التعلم. يتم استخدام زيادة البيانات لمعالجة عدم توازن الفئات، مما يضمن قدرة النموذج على التعميم.

إن جوهر النظام هو شبكة عصبية تلافيفية خفيفة الوزن (lightweight CNN) ، تم تحسينها لمجموعات بيانات التدريب الصغيرة والتكلفة الحسابية المنخفضة، مما يجعلها مناسبة للأجهزة ذات الموارد المحدودة وتطبيقات الوقت الحقيقي. تتم مقارنة أداء التصنيف للنموذج المقترح بنماذج الشبكات العصبية التلافيفية (CNN) المستخدمة على نطاق واسع، بما في ذلك DenseNet و ResNet و InceptionV3 وVGG16 وVGG19 . وتوضح النتائج تفوق نموذج سي إن إن الخفيف الوزن من حيث تكلفة الحوسبة والتعميم، مما يشير إلى آفاق واعدة للتطبيقات العملية. يساهم هذا البحث في تطوير نظام التشخيص ودعم اتخاذ القرار الطبي لكوفيد-19 من خلال تقنيات التعلم العميق، لا سيما في المناطق التي يكون فيها اختبار تفاعل البوليميراز المتسلسل محدودًا أو لا يمكن الوصول إليه بسهولة. من خلال تقديم بديل أسرع وموثوق به وفعال من حيث التكلفة لفحص تفاعل البوليميراز المتسلسل، يسهم هذا البحث في تطوير تشخيص كوفيد-19 ودعم القرارات الطبية من خلال تقنيات التعلم العميق، لا سيما في المناطق ذات الموارد المحدودة.

كلمات مفتاحية: كوفيد-19، تصوير الصدر بالأشعة السينية، التصوير الطبي، الشبكة العصبية التلافيفية، تجزئة الرئتين، تحسين الضوء.

« Détection et compréhension des objets dans une séquence d'images : Application à l'imagerie médicale »

Résumé :

Cette étude explore l'utilisation de la détection et de la compréhension d'objets dans une série d'images pour diagnostiquer la COVID-19, en plus d'analyser des radiographies pulmonaires individuelles. En utilisant des réseaux de neurones convolutionnels (CNN) avec des étapes de prétraitement adéquates, le modèle extrait des motifs et des dépendances temporelles à partir de données d'imagerie séquentielle, aidant ainsi à surveiller la progression de la maladie, l'efficacité du traitement et à prédire les résultats. La pandémie de COVID-19, originaire de Wuhan, en Chine, en 2019, s'est propagée à plus de 200 pays, imposant un fardeau important aux systèmes de santé mondiaux et à l'économie. Le besoin urgent de méthodes de diagnostic plus rapides et plus accessibles est devenu évident, notamment dans les régions à ressources limitées. La détection précoce peut aider à réduire les taux de mortalité et à prévenir la transmission de la maladie. Alors que les tests de réaction en chaîne par polymérase inverse (RT-PCR) ont été largement utilisés comme méthode fiable pour détecter le virus, ils présentent des inconvénients tels que le temps, le coût et le risque pour les travailleurs de la santé en raison du contact étroit avec les patients. En réponse à ces défis, des techniques d'imagerie médicale comme les radiographies pulmonaires ont été utilisées pour le dépistage de la COVID-19 en raison de leur rapidité, de leur sécurité et de leur disponibilité plus large par rapport aux méthodes traditionnelles. L'intégration de l'intelligence artificielle (IA) dans le traitement d'images a montré des résultats prometteurs pour distinguer avec précision les radiographies thoraciques normales et malades, aidant ainsi au diagnostic précoce et à la gestion des patients. Dans cette étude, nous proposons un modèle d'apprentissage en profondeur pour analyser les radiographies pulmonaires de la COVID-19, visant à résoudre les limitations des tests de PCR et à améliorer l'efficacité du diagnostic. Les défis incluent des données d'entraînement limitées, impactant la généralisation du modèle. La méthode implique un prétraitement par Simplest Color Balance, une segmentation pulmonaire utilisant U-Net, une augmentation de données pour le déséquilibre des classes et un CNN léger pour un calcul efficace. La méthode proposée comprend une étape de prétraitement pour améliorer la qualité de l'image en utilisant Simplest Color Balance, suivie de la segmentation pulmonaire à l'aide de U-Net pour éliminer les données non pertinentes et améliorer l'apprentissage. L'augmentation de données est utilisée pour traiter le déséquilibre des classes, garantissant la capacité de généralisation du modèle. Le cœur du système est un réseau de neurones convolutionnels (CNN) léger, optimisé pour de petits ensembles de données d'entraînement et un faible coût de calcul, le rendant adapté aux appareils à ressources limitées et aux applications en temps réel. La performance de classification du modèle proposé est comparée à celle de modèles CNN largement utilisés, notamment DenseNet, ResNet, InceptionV3, VGG16 et VGG19. Les résultats illustrent la supériorité du modèle CNN léger en termes de coût de calcul et de généralisation, indiquant des perspectives prometteuses pour des mises en œuvre pratiques. Cette recherche contribue à faire progresser le diagnostic et le système de soutien à la décision médicale de la COVID-19 grâce à des techniques d'apprentissage en profondeur, en particulier dans les régions où les tests de PCR sont limités ou difficilement accessibles. En offrant une alternative plus rapide, fiable et économique à la PCR, cette recherche fait progresser le diagnostic de la COVID-19 et le soutien à la décision médicale grâce à des techniques d'apprentissage en profondeur, notamment dans les régions à ressources limitées..

Mots clés : Covid-19, Chest xrays, medical imaging, Convolutional neural network, Lungs segmentation, Light enhancement.

« Detection and Understanding of Objects in an Image Sequence : Application to Medical Imaging»

Abstract :

This study explores the use of object detection and understanding in a series of images to diagnose COVID-19, alongside analyzing individual chest X-rays. By employing convolutional neural networks (CNNs) with adequate preprocessing steps, the model extracts patterns and temporal dependencies from sequential imaging data, aiding in monitoring disease progression, treatment effectiveness, and predicting outcomes. The COVID-19 pandemic, originating in Wuhan, China, in 2019, has spread to over 200 countries, imposing a significant burden on global health systems and the economy. The pressing need for quicker and more accessible diagnostic methods has become apparent, especially in resource-constrained regions. Early detection can aid in reducing mortality rates and preventing disease transmission. While reverse transcription polymerase chain reaction (RT-PCR) tests have been widely utilized as a reliable method for detecting the virus, they have drawbacks such as being time-consuming, costly, and risky for healthcare workers due to close patient contact. In response to these challenges, medical imaging techniques like X-rays have been employed for COVID-19 screening due to their speed, safety, and wider availability compared to traditional methods. Integrating artificial intelligence (AI) into image processing has shown promising results in accurately distinguishing between normal and diseased chest radiographs, aiding in early diagnosis and patient management. In this study, we propose a deep learning model to analyze COVID-19 chest X-rays, aiming to address the limitations of PCR testing and enhance diagnostic efficiency. Challenges include limited training data, impacting model generalization. The method involves Simplest Color Balance preprocessing, lung segmentation using U-Net, data augmentation for class imbalance, and a lightweight CNN for efficient computation. The proposed method comprises a preprocessing step to enhance image quality using Simplest Color Balance, followed by lung segmentation using U-Net to eliminate irrelevant data and enhance learning. Data augmentation is employed to tackle class imbalance, ensuring the model's generalization capability. The core of the system is a lightweight convolutional neural network (CNN), optimized for small training datasets and low computation cost, rendering it suitable for resource-constrained devices and real-time applications. The classification performance of the proposed model is compared with widely-used CNN models, including DenseNet, ResNet, InceptionV3, VGG16, and VGG19. The results illustrate the superiority of the lightweight CNN model in terms of computation cost and generalization, indicating promising prospects for practical implementations. This research contributes to advancing the diagnosis and medical decision support system of COVID-19 through deep learning techniques, particularly in regions where PCR testing is limited or not readily accessible. By offering a faster, reliable, and cost-effective alternative to PCR, this research advances COVID-19 diagnosis and medical decision support through deep learning techniques, especially in resource-constrained regions.

Key words : Covid-19, Chest xrays, medical imaging, Convolutional neural network, Lungs segmentation, Light enhancement.

Contents

List of Figures

List of Tables

General Introduction

The detection of COVID-19 in X-ray images is a critical and challenging problem affecting millions of individuals worldwide, with significant implications for healthcare systems and individual patients. The COVID-19 pandemic, which emerged in Wuhan, China in 2019, has rapidly spread across the globe, placing immense strain on healthcare resources and economies worldwide. As a result, there is an urgent need for faster and more accessible diagnostic methods, particularly in resource-constrained regions, to aid in reducing mortality rates and preventing disease transmission. Our number one concern was to a solution designed to address this challenge by offering an affordable, portable, and efficient method suitable for everyday use. Traditional diagnostic methods such as reverse transcription polymerase chain reaction (RT-PCR) tests, while reliable, are hindered by factors such as time consumption, costliness, and the risk of exposure for healthcare workers due to close patient contact. In response to these challenges, medical imaging techniques like X-rays have emerged as valuable tools for COVID-19 screening, offering advantages in terms of speed, safety, and wider availability compared to traditional methods. By integrating artificial intelligence (AI) into image processing, promising results have been achieved in accurately distinguishing between normal and diseased chest radiographs, thereby facilitating early diagnosis and patient management.

This thesis presents a significant *contribution* in the form of a novel approach to COVID-19 diagnosis in X-ray images by employing deep learning techniques, specifically convolutional neural networks (CNNs). The proposed solution involves many *contributions*: such us **(1)** the development of a custom dataset comprising confirmed COVID-19 sub-classes, which is used to **(2)** train and fine-tune a lightweight convolutional neural network (lightweight CNN), optimized for small training datasets and low computation cost, rendering it suitable for resource-constrained devices and real-time applications, designed to achieve high generalization performance. This characteristic of being lightweight holds significant practical implications, as it enables the proposed model to be deployed on devices with limited processing capabilities, such as edge devices and single-board computers, facilitating real-time operation. Preprocessing techniques such as **(3)** Simplest Color Balance, that aimed at enhancing image quality through the application of the color balance algorithm. This step ensures improved readability of tissues without altering the original data, and **(4)** subsequently, lung segmentation using U-Net is integrated to eliminate irrelevant data and enhance learning, while **(5)** data augmentation is employed to address the effects of class imbalance and ensure the model's generalization capability.

In term of System's Evaluation, Extensive testing across various external datasets has been

conducted to assess the system's performance, reliability, and generalization ability. The classification performance of the proposed model is compared with widely-used CNN models, including DenseNet, ResNet, InceptionV3, VGG16, and VGG19. These models differ in terms of design, number of parameters, and depth. These factors allow for a comparison of models interchangeably with the proposed model. The results demonstrate the real-time capability and reliability of the system in accurately detecting COVID-19 and non-COVID classes, thereby providing timely decision support. Moreover, our system exhibits high portability and efficiency, rendering it practical and effective for real-time and embedded systems. The lightweight CNN model, optimized for small training datasets and low computation cost, exhibits superior performance in terms of computation cost and generalization compared to traditional CNN models.

Overall, this study offers a comprehensive and innovative solution to the challenge of COVID-19 detection in X-ray images, contributing to the advancement of medical decision support systems in the fight against the pandemic, particularly in regions where traditional diagnostic methods are limited or inaccessible.

Thesis Structure

The thesis is structured into four chapters:

- **Chapter 1: Introduction to AI-Based Solutions for real time objects Detection** This chapter compares human vision with AI capabilities, explores traditional solutions, and highlights the potential benefits of AI. It provides an overview of AI and focuses on deep learning for developing effective solutions.

- **Chapter 2: CNNs in Depth** This chapter delves into the intricacies of CNNs, detailing each layer used and discussing the selection of the most appropriate functions.

- **Chapter 3: System Architecture and Training Process, for our The New approach based on light enhancement and real-time dual CNN for classification of COVID-19 X-ray images** Focusing on the proposed model's architecture and the training process, this chapter highlights the critical steps in developing an AI system for accurately detecting the presence of Covid or non-Covid in X-ray images of suspected patients. It also examines the project's development environment, including hardware and software tools, addressing challenges, and highlighting outcomes and achievements.

and we finish with a general conclusion and perspectives.

Chapter 1

Introduction to AI Based Solutions for real time objects Detection

1.1 Introduction

In the age of digitization and automation, by the proliferation of visual data and the unceasing evolution of artificial intelligence, real-time object detection and comprehension in image sequences have emerged as a fundamental challenge at the intersection of computer vision, artificial intelligence, and real-time processing, and addresses the core task of endowing machines with the ability to not only identify objects within a stream of images but also to rapidly and accurately interpret and comprehend the contextual nuances of these objects, close to how humans perceive and understand the world.

In the realm of computer vision, the ability to understand and interpret visual data in real time has ushered in a new era of technological innovation. The convergence of deep learning with image analysis has made it possible to achieve tasks that were once considered science fiction.

This capability holds profound implications for a large amount of applications, ranging from autonomous vehicles navigating complex environments to surveillance systems monitoring public spaces, medical imaging and disease detection, and from robotics interacting with their surroundings to augmented reality enhancing human-computer interactions.

The need for real-time object detection and comprehension is driven by the ever-increasing volume of visual data generated daily, whether in the form of surveillance footage, streaming video, or images captured by devices, such as smartphones and cameras. Traditional computer vision systems, while capable of identifying objects within images, often fall short when it comes to processing vast streams of data in real time. This limitation has sparked intensive research into innovative techniques and algorithms aimed at addressing the twin goals of accuracy and speed.

Real-time object detection, a field that has made significant strides, has historically been challenged by the computational intensity of processing image sequences at video frame rates. Conventional computer vision techniques, while proficient in static image analysis, struggle to

meet the demands of real-time applications.

This thesis endeavors to explore, develop, and evaluate state-of-the-art methodologies for real-time object detection and comprehension in image sequences. The foundation of this research is rooted in the advancements of deep learning, a transformative technology that has revolutionized computer vision. The emergence of deep learning, particularly convolutional neural networks (CNNs), has been a transformative catalyst, enabling real-time processing through their capacity for parallelization, adaptability, and hierarchical feature representation.

This chapter delves into the exciting world of real-time object detection, segmentation, and classification using deep learning, where algorithms not only perceive but also make intelligent decisions about the visual world. So in this chapter, we will focus on methodologies, technologies, and techniques that facilitate real-time object detection and comprehension in image sequences, by exploring various popular models of object classification, detection and segmentation such as R-CNN, SSD, RetinaNet, and YOLO...etc. Additionally, We will discuss the challenges and the proposed solutions to bridge the gap between accuracy and speed, We will also discuss some of the benchmark datasets that have been used to train and evaluate machine learning models. Finally, we will provide a taxonomy and summaries the chapter, and introduce the next chapter.

1.2 The importance of real-time object detection and comprehension in various applications

In recent years, Artificial Intelligence (AI) based solutions for object detection, classification and comprehension in image sequences play a crucial role in a wide range of applications across various domains, and have gained immense popularity and transformed a wide range of industries. These solutions have revolutionized the way we perceive, interact with, and automate tasks related to the visual world.

Real-time Object detection and classification powered by AI enable computers to recognize and categorize objects in images and videos, making them valuable tools in fields such as :

- Enhanced Safety and Security: In applications such as surveillance and security, real-time object detection can quickly identify potential threats, intruders, or unusual activities. This capability is vital for the safety of individuals, facilities, and public spaces.

- Autonomous Vehicles: Autonomous cars and drones rely heavily on real-time object detection to perceive their surroundings. It enables them to navigate and make real-time decisions to avoid obstacles, pedestrians, and other vehicles, contributing to safer and more efficient transportation.

- Human-Computer Interaction: Real-time object detection is central to human-computer interaction systems. It allows for gesture recognition, hand tracking, and other natural

interfaces, making technology more accessible and user-friendly.

- Manufacturing and Quality Control: In manufacturing, real-time object detection ensures quality control by identifying defects, ensuring proper assembly, and monitoring production processes. This leads to increased efficiency and cost savings.

- Medical Imaging: In the medical field, real-time object detection aids in identifying anomalies in medical images, supporting early diagnosis and intervention. For instance, it can help detect tumors, anomalies in X-rays, or abnormalities in real-time during surgery.

- Agriculture: Precision agriculture benefits from object detection to monitor crop health, detect pests, and automate tasks like weeding and harvesting. This technology helps optimize crop yields and reduce resource usage.

- Retail and E-commerce: In the retail industry, real-time object detection is used for inventory management, monitoring customer behavior, and creating personalized shopping experiences. It can also enable cashierless stores, improving convenience for shoppers.

- Search and Recommendation Systems: In online platforms, object detection and comprehension are used to analyze and tag images and videos. This helps users discover relevant content, facilitates content moderation, and enhances the overall user experience.

- Environmental Monitoring: Environmental scientists use real-time object detection to monitor and study wildlife behavior, track climate change effects, and assess the impact of human activities on ecosystems.

- Smart Cities: Real-time object detection can be employed in smart city initiatives for traffic management, waste collection, public safety, and urban planning. It enables cities to become more efficient and sustainable.

- Augmented and Virtual Reality: In AR and VR applications, real-time object detection is essential for creating immersive experiences by recognizing real-world objects and enhancing the virtual environment.

- Accessibility and Inclusion: Real-time object detection technology can be used to assist individuals with disabilities, such as helping the visually impaired navigate their surroundings or aiding those with mobility challenges.

In summary, real-time object detection and comprehension are at the heart of numerous applications that improve safety, efficiency, convenience, and overall quality of life. These technologies have the potential to transform industries and create innovative solutions for complex challenges, making them a significant area of research and development.

1.3 Human Vision System vs Computer Vision systems

1.3.1 Anatomy and Pathway of the Human Eye

understanding the human visual system is crucial for developing effective computer vision systems. The human visual system is incredibly complex and efficient, and it serves as a model for designing and improving artificial vision systems.

By understanding the principles of the human visual system, researchers can design computer vision algorithms and models that mimic its efficiency and effectiveness. This can lead to advancements in various computer vision applications, including object recognition, image classification, and scene understanding.

1.3.1.1 The Anatomy of the eye

The anatomy and function of the human eye play a crucial role in our ability to perceive the world around us, and understanding them is essential to developing solutions for visual impairment. In this section, we will provide an overview of the structure and function of the eye, including its various components and how they work together to process light and transmit neural signals to the brain.
The human eye is a complex sensory organ responsible for vision. It consists of various anatomical structures, each with specific functions that contribute to the overall process of visual perception. Light entering the eye is focused, transduced into electrical signals, and transmitted to the brain, where the visual information is processed and interpreted to provide us with the sense of sight. Here is an overview of the anatomy and function of the human eye:

- **1. Cornea:** It acts as a protective barrier and is primarily responsible for refracting (bending) incoming light.
- **2. Iris:** The iris regulates the amount of light entering the eye by adjusting the size of the pupil. It contracts in bright light and dilates in dim light.
- **3. Pupil:** The pupil controls the amount of light that enters the eye. It dilates to allow more light in low-light conditions and constricts in bright light.

- **4. Lens:** The lens further refracts light and focuses it onto the retina. It can change shape (accommodation) to allow focusing on objects at varying distances.
- **5. Retina:** The retina contains photoreceptor cells (rods and cones) that capture and transduce light into electrical signals. These signals are sent to the brain via the optic nerve.
- **6. Photoreceptor Cells:** Rods are responsible for vision in low-light conditions and contribute to black-and-white vision. Cones are responsible for color vision and function best in bright conditions.
- **7. Optic Nerve:** The optic nerve carries visual information from the retina to the brain for further processing.
- **8. Optic Chiasm:** It allows the integration of visual information from both eyes and ensures that each hemisphere of the brain receives input from both visual fields.
- **9. Lateral Geniculate Nucleus (LGN):** The LGN further processes visual information before sending it to the primary visual cortex.
- **10. Visual Cortex:** The primary visual cortex processes visual information received from the LGN, allowing for perception of the visual world, object recognition, and interpretation of colors, shapes, and motion.
- **11. Extraocular Muscles:** These muscles control eye movement, enabling tracking and convergence for accurate focusing on objects.
- **12. Aqueous and Vitreous Humors:** These fluids maintain the shape and pressure of the eye and help with focusing light.

The eye's structure encompasses the above mentioned components and elements constituting the organ crucial for vision. It is a sophisticated assembly enabling the capture of light from the surroundings and its transformation into neural signals relayed to the brain. With a modest volume of 6.5 cm3, a weight of around 7 grams, and a spherical form measuring approximately 24 mm in diameter, the eye is augmented at its anterior by a semispherical extension with an 8 mm radius. Comprising 13 distinct components, each fulfilling a designated role, the eye's anatomy is elucidated in Figure 1.1.

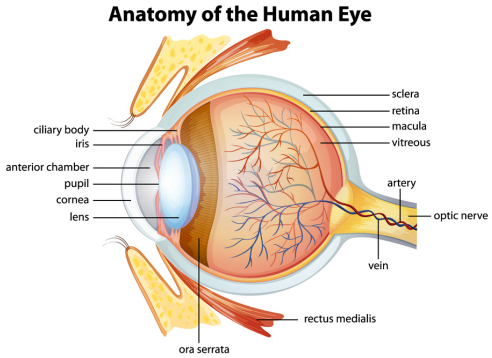


Figure 1.1: human eye anatomy [1]

1.3.1.2 The visual pathway

The pathway of vision, also known as the visual pathway, describes the sequence of events that occur from the moment light enters the eye until visual information is processed in the brain. The pathway of vision is a complex and highly organized process that allows us to perceive and understand the visual world. It involves multiple stages of information processing and interaction between various parts of the eye and the brain:

(a) **Light Entering the Eye:** The process begins when external light from the environment enters the eye through the cornea, which is the clear, curved front surface of the eye. The cornea helps to focus the light onto the next structure in the pathway.

(b) **Pupil and Iris:** The amount of light entering the eye is regulated by the iris, which is the colored part of the eye. The iris can contract or dilate to adjust the size of the pupil, which is the black center of the eye. In bright conditions, the pupil contracts to limit the amount of light, while in dim conditions, it dilates to allow more light to enter.

(c) **Lens:** The light then passes through the lens, which further refracts (bends) the light to ensure that it focuses onto the retina at the back of the eye. The lens adjusts its shape to accommodate varying distances, allowing us to focus on objects at different distances.

(d) **Retina:** The retina is the neural tissue at the back of the eye that contains photoreceptor cells, namely, rods and cones. These cells are responsible for capturing and transducing light into electrical signals. Rods are sensitive to low light and contribute to black-and-white vision, while cones are responsible for color vision and function best in brighter conditions.

(e) **Phototransduction:** When light strikes photoreceptor cells in the retina, it triggers a process called phototransduction. Photopigments in the rods and cones change shape in response to light, initiating an electrical signal. These signals are then processed by the various layers of cells in the retina.

(f) **Bipolar Cells and Ganglion Cells:** After being processed by various layers of interneurons in the retina, the electrical signals are transmitted from photoreceptor cells

to bipolar cells and then to ganglion cells. Ganglion cells are the output neurons of the retina and send the visual information to the brain via the optic nerve.

(g) **Optic Nerve:** The ganglion cells' axons converge to form the optic nerve, which carries the visual information from the eye to the brain. The optic nerve exits the eye at the optic disc, creating a blind spot in the visual field.

(h) **Optic Chiasm:** At the base of the brain, the optic nerves from both eyes partially cross over at a structure called the optic chiasm. This crossover allows for the integration of visual information from both hemispheres of the brain.

(i) **Optic Tract:** The optic tracts, which consist of nerve fibers that originated from the optic chiasm, continue to transmit visual information to different parts of the brain.

(j) **Lateral Geniculate Nucleus (LGN):** Visual information is relayed to the thalamus, specifically the lateral geniculate nucleus (LGN). The LGN further processes the visual information and sends it to the visual cortex in the brain.

(k) **Visual Cortex:** The primary visual cortex, located in the occipital lobe at the back of the brain, processes the visual information received from the LGN. The visual cortex interprets this information, allowing us to perceive the visual world, recognize objects, and perceive colors, shapes, and motion.

(l) **Higher-Order Processing:** Beyond the primary visual cortex, visual information is transmitted to higher-order visual areas and other parts of the brain, contributing to the perception of complex visual scenes, object recognition, and higher-level cognitive processes.

we can resume it, When light enters the eye, it stimulates pigments found in the photoreceptor cells, leading to their activation and generation of electrical impulses. These impulses then travel through a network of neurons to reach the ganglion cells, which in turn transmit them to the brain's visual cortex. The visual cortex processes these signals, allowing us to perceive the object.

1.3.1.3 The field of vision

The field of vision is divided between the two eyes, with each eye receiving information from half of the visual field. The field of vision, also known as the visual field or field of view, is the entire extent of the observable world that an individual can see at any given moment without moving their eyes or head. It encompasses both the central vision, which provides detailed and high-resolution visual information, and the peripheral vision, which offers a broader but less detailed view of the surroundings. The field of vision is typically measured in degrees and is an essential concept in the study of human vision and optics. Differences in the peripheral parts of each eye's visual field contribute to depth perception or the ability to see objects in

three dimensions. Depth perception is crucial in accurately judging distances and estimating the sizes and dimensions of objects (figure 1.2).

The visual field is divided between the two eyes, with each eye receiving information from half of the field of view. This area, also referred to as the field of vision or visual field, encompasses the entire observable world that an individual can see without moving their eyes or head. It includes central vision, which provides detailed and high-resolution visual information, as well as peripheral vision, offering a broader but less detailed view of the surroundings. Typically measured in degrees, the field of vision is a fundamental concept in the study of human vision and optics. Variances in the peripheral areas of each eye's visual field contribute to depth perception, enabling the perception of objects in three dimensions. Depth perception is crucial for accurately gauging distances and estimating the sizes and dimensions of objects (see Figure 1.2).

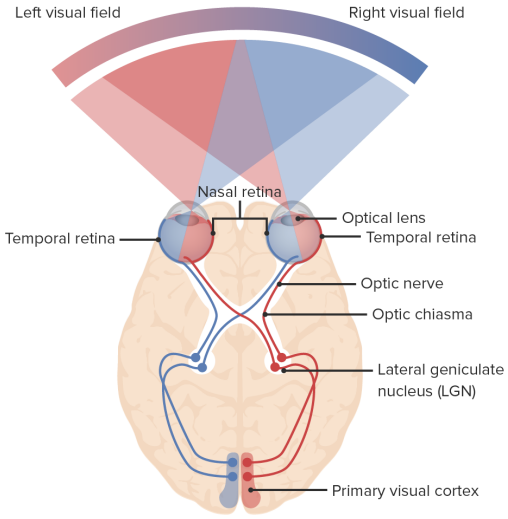


Figure 1.2: The visual projection pathway and field.

1.3.2 The Potential of the Human Visual System

The human visual system is a highly sophisticated and complex sensory system that allows us to *perceive*, *interpret*, and *understand* the visual world. It achieves this by:

(a) Interpreting and comprehending complex visual scenes in real-time.

(b) Detecting and recognizing a wide array of visual features, including colors, shapes, textures, and movements.

(c) Adapting effectively to diverse lighting conditions, viewing distances, and angles.

(d) Easily recognizing familiar objects and faces, even when they are partially obscured or viewed from different perspectives.

(e) Executing complex visual tasks, such as reading and recognizing text, with remarkable ease and without conscious effort.

It also encompasses a wide range of capabilities that contribute to our ability to see and make sense of our surroundings. thses capabilities are :

Visual Perception, Visual Acuity, Color Vision, Depth Perception, Pattern Recognition, Motion Detection, Low-Light Vision, Peripheral Vision, Color Constancy, Emotional Recognition, Visual Memory, Gestalt Principles, Adaptation to Context, Top-Down Processing, Visual Attention, Holistic Processing.

These capabilities of the human visual system are fundamental to our daily lives, enabling us to navigate our environment, engage in social interactions, and gather information from the visual world in a meaningful and nuanced way. Until now, scientists are striving to emulate human vision capabilities in order to develop a computer vision system, the figure 1.3.

1.3.3 Human Vision and Machine Vision: A Comparative Analysis

The Human vision and The Machine vision represent two distinct approaches to perceiving and processing visual information. Human vision, as a biological sensory system, is dynamic, adaptable, and versatile, with the ability to recognize patterns, objects, and scenes in real-time. In contrast, the machine vision system, is an artificial system, and is static and relies on pre-defined algorithms and parameters to capture and process visual data. Machine vision excels in tasks requiring high accuracy, precision, speed, and consistency, making it ideal for applications like quality control in manufacturing. Human vision, on the other hand, offers adaptability, contextual understanding, and the ability to make subjective judgments, making it suitable for tasks where cognitive reasoning and emotional assessment are essential.

Aspect	Human Vision	Machine Vision (Traditional)	Computer Vision (Deep Learning)
Biological vs. Artificial Neural Networks	Relies on the complex human brain and biological neurons	Utilizes pre-defined algorithms and image processing techniques	Utilizes artificial neural networks inspired by the human brain's structure.
Learning Mechanisms	Acquired through years of exposure and learning	Utilizes pre-defined algorithms and parameter settings	Requires training data and deep learning algorithms for learning and pattern recognition.
Subjectivity vs. Objectivity	Inherently subjective, influenced by individual experiences and emotions	Objective, particularly in traditional systems for specific tasks	Designed for objectivity, providing consistent and data-driven assessments.
Processing Speed and Data Volume	Quick adaptation but may not handle large data volumes efficiently	Capable of processing data but not as rapidly or efficiently as deep learning	Processes large datasets rapidly with high accuracy, suitable for real-time tasks.
Adaptability and Generalization	Highly adaptable and capable of generalization	Limited adaptability and generalization in traditional systems	Adapts and generalizes to some extent but may require retraining for novel situations or objects.

Tableau 1.1: Comparison of Human Vision, Machine Vision (Traditional), and Computer Vision (Deep Learning)

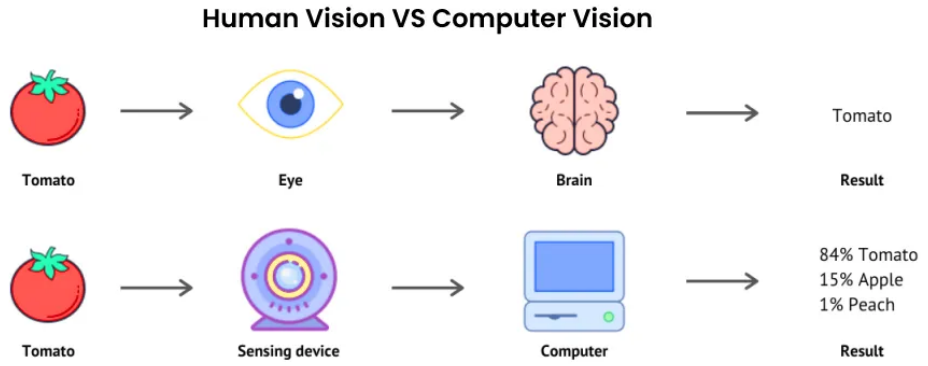


Figure 1.3: Human vision VS Computer vision

1.3.4 Computer/machine Vision

Computer vision, also known as machine vision, is a multidisciplinary field of artificial intelligence that focuses on enabling computers and machines to interpret and understand visual information from the world, much like human vision. It involves the development of algorithms and technologies to process, analyze, and make sense of digital images and videos, allowing machines to recognize objects, and patterns, and perform tasks related to visual perception.

- The term "visual machine" refers to machine learning algorithms and computer vision systems designed to process and understand visual data extracted from digital images and videos.
- These systems possess the ability to recognize and categorize various visual components or patterns, such as colors, shapes, faces, objects, and text, utilizing sophisticated mathematical algorithms and neural networks. The accuracy and reliability of such systems rely on the quality and quantity of data they are trained with, as well as the specific algorithm and architecture employed.
- These systems can execute complex visual tasks, including object classification, detection, and segmentation, with remarkable precision, making them valuable in numerous applications like autonomous vehicles, medical imaging, and surveillance systems.
- However, these machine vision systems have limitations. For instance, they may encounter difficulties with low-quality or noisy images and may also exhibit biases or errors in their analysis and interpretation of visual data.

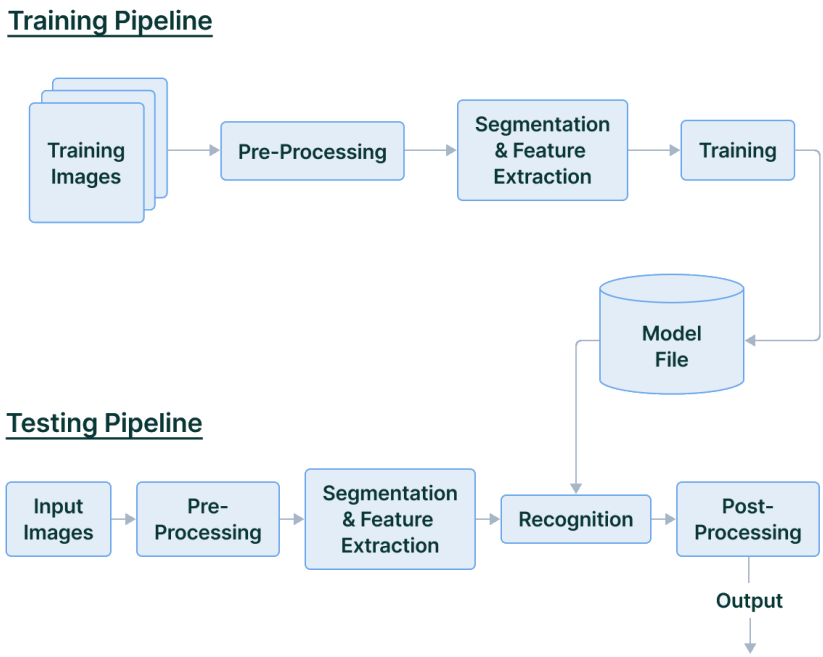


Figure 1.4: The Computer vision pipeline for object recognition

1.3.4.1 Challenges in Visual Machine Systems

In contrast to the human visual system (depicted in Figure 1.3), the visual machine operates within predefined parameters established based on its training data. If the training data is biased or incomplete, the machine's real-world performance can suffer. Furthermore, the machine's capability is confined to the specific tasks it was trained for; it may struggle to adapt to new scenarios or recognize unfamiliar objects. Another limitation lies in the machine's inability to reason and contextualize information as comprehensively as humans. While it may be able to recognize objects in images, it often falls short of understanding the relationships between these objects and grasping the overall meaning of the image, potentially leading to errors and misinterpretations. Additionally, the computational power and resources available to the visual machine can pose limitations, as more complex tasks or extensive datasets may necessitate more powerful hardware and extended training periods, incurring considerable costs or logistical challenges.

We can detail the limitations of the visual machine in:

- Data Dependency: These systems heavily rely on the quality and quantity of data they are trained on. Inadequate or biased training data can result in suboptimal performance and potential biases in recognition.
- Computational Requirements: Complex computer vision tasks can demand significant computational resources, making real-time processing challenging without powerful hardware.
- Environmental Variability: Visual machines may struggle in scenarios with rapidly changing lighting conditions, weather, or unfamiliar environments, which can affect their accuracy.
- Ambiguity and Context: Understanding context and dealing with ambiguous situations can be challenging for visual machines. They may misinterpret certain scenarios that a human would easily understand.
- Privacy Concerns: In applications like surveillance, there are privacy concerns related to the constant monitoring and analysis of visual data. Striking a balance between security and privacy is an ongoing challenge.
- Ethical Issues: The use of visual machines can raise ethical questions, such as bias in algorithms, data privacy, and surveillance ethics, which need to be addressed for responsible implementation.
- Limited Generalization: Visual machines may excel in specific tasks they are trained for but can struggle with tasks outside their training domain. Achieving broad generalization remains a challenge.

- Security Vulnerabilities: Like any technology, computer vision systems are vulnerable to hacking and adversarial attacks, potentially compromising their accuracy and reliability.
- Cost of Implementation: Developing and deploying advanced visual machines can be costly, limiting their accessibility in some applications and industries.
- Regulatory and Legal Challenges: Evolving regulations and legal frameworks surrounding the use of computer vision technology can pose challenges for its adoption and deployment in various domains.

Addressing these limitations is an ongoing area of research and development in the field of computer vision, and one of the most important solutions is the use of Artificial Intelligence Algorithm such as the Convolutional neural networks (CNNs).

CNNs are the milestone of the modern deeplearning revolution.

The success of CNNs has led to breakthroughs in various applications, including image classification (e.g., the ImageNet challenge), object detection (e.g., with architectures like YOLO and Faster R-CNN), image segmentation, facial recognition, medical image analysis, and more. before explaining CNNs, we most explain AI and Deep Learning and the most important features of each field.

1.4 Deep Learning in AI

1.4.1 Artificial intelligence

Artificial Intelligence (AI) refers to the development of computer systems and software that can perform tasks that typically require human intelligence. These tasks can encompass a wide range of activities, from simple, rule-based processes to complex, decision-making tasks. AI systems aim to mimic, replicate, or augment human cognitive functions, such as machine learning, problem-solving, understanding natural language, computer vision, robotics, expert systems, and recognizing patterns.

AI is a field of computer science that aims to develop intelligent systems capable of performing tasks that typically require human-level intelligence. It involves the study and development of algorithms and models that enable machines to learn from and make decisions based on data, reason, and plan, and communicate using natural language. [20] (figure 1.5).

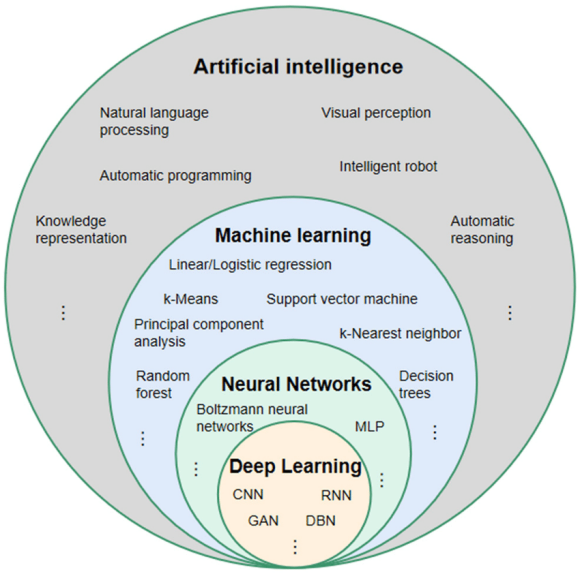


Figure 1.5: Different subdomain AI, ML and DL

The key aspects of artificial intelligence include:

(a) **Machine Learning:** Machine learning is a subset of AI that involves training algorithms to recognize patterns and make predictions based on data. This is often used in applications like image recognition, natural language processing, and recommendation systems.

(b) **Deep Learning:** Deep learning is a specific type of machine learning that utilizes neural networks with many layers to process and interpret data. Deep learning has led to significant advancements in tasks like image and speech recognition.

(c) **Natural Language Processing (NLP):** NLP focuses on enabling computers to understand, interpret, and generate human language. It's used in chatbots, language translation, and text analysis.

And other subdomains like:

- **Computer Vision:** Computer vision allows machines to interpret and understand visual information from the world, making it applicable in tasks like facial recognition and autonomous vehicles.
- **Expert Systems:** These are AI systems that mimic the decision-making capabilities of a human expert in a specific domain. They use rules and knowledge bases to make informed decisions.
- **Robotics:** AI plays a vital role in the field of robotics, enabling robots to perceive their environment and make autonomous decisions.
- **Reinforcement Learning:** In reinforcement learning, AI agents learn by interacting with their environment and receiving feedback in the form of rewards or penalties. It's used in areas like game-playing AI and autonomous control systems.

1.4.2 Machine learning

Machine learning is a subfield of artificial intelligence (AI) that focuses on developing algorithms and models that enable computers to learn from and make predictions or decisions based on data. It's essentially the science of teaching machines to improve their performance on a specific task through experience, without being explicitly programmed for that task. It made programming easier by taking in just training data as inputs, which is combined of pairs of data (features) and what their outputs should be (labels). ML figures out the patterns or the rules by itself, but the main goal in ML is to raise the accuracy as high as possible which means it still can make mistakes. However, to raise the accuracy, ML needs a massive amount of inputs as examples to train a good model that than predict the best possible output for new unknown data.

Machine learning can be supervised, unsupervised, or semi-supervised, and can be used for a wide range of applications, including image and speech recognition, natural language processing, fraud detection, recommendation systems, and autonomous vehicles. [21–23]

Tom Mitchell. [23] provided a formal definition of the algorithms studied in the machine-learning field: *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."*

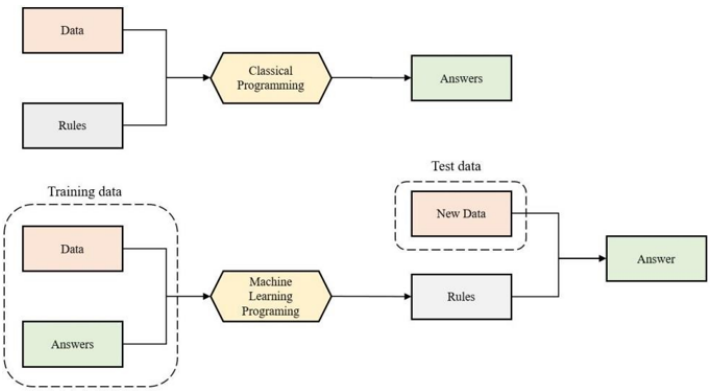


Figure 1.6: Difference between Classical Programming and ML Programming learning steps

1.4.2.1 Differences Between AI and Machine Learning :

Artificial Intelligence	Machine Learning
AI involves computer systems performing tasks that typically require human intelligence.	ML uses data and algorithms to learn and improve autonomously.
AI aims to emulate human thinking, learning, and decision-making processes.	ML focuses on acquiring knowledge through self-generated algorithms.
AI's primary focus is on decision-making.	ML involves using algorithms for computers to learn and improve based on experience.
Examples of AI include widely used software such as Siri, Google Translate, Google Assistant, and chatbots.	Examples of ML can be found in everyday applications like recommendation engines, Facebook friend suggestions, and traffic alerts.

Tableau 1.2: Differences between AI and ML

This Figure 1.7 illustrates the difference between the human learning (HL) and the Machine Learning (ML).

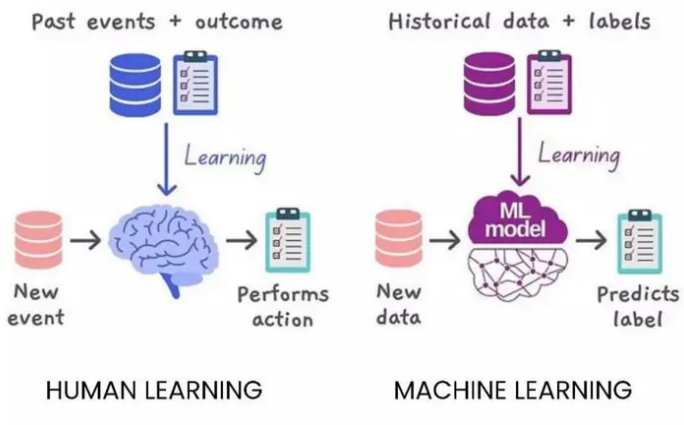


Figure 1.7: HL V ML

1.4.3 Types of Machine Learning

we have various types of machine learning

1.4.3.1 Supervised Learning:

In supervised learning, the model is trained using labeled data, meaning it learns to predict a specific output (e.g., class labels or numerical values) from input data. Supervised learning is the most applicable type of learning; its algorithms build a mathematical model of a set of data that contains both the inputs and their desired outputs. Called the Training data, where each training example is represented by an array or vector, sometimes called a feature vector, and

the full training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn an inferred function that can be used to predict the output associated with new unknown inputs [13]. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way. The computer "learns" from the observations, when exposed to more observations, the computer than improves its predictive performance. Specifically, a supervised learning algorithm takes a known set of input data and known responses to the data (output), and trains a model to generate reasonable predictions for the response to new data. (figure 1.8).

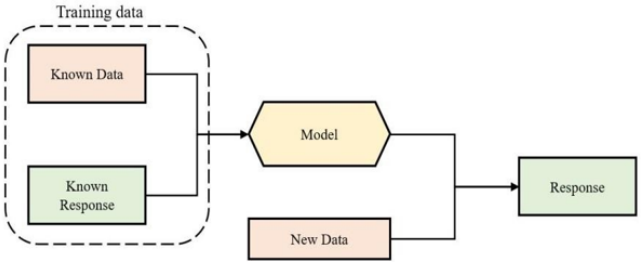


Figure 1.8: Supervised learning steps

1.4.3.2 Unsupervised Learning:

Unsupervised learning (figure 1.9) involves finding patterns or structures in data without labeled outputs. Clustering and dimensionality reduction are common unsupervised learning tasks. Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points, without having previously an idea of what they can be or how many they should be, by figuring out the similarities between the given data with no human intervention.

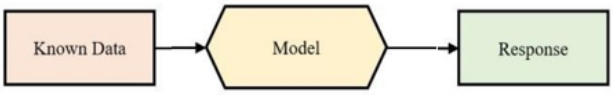


Figure 1.9: Unsupervised learning steps

1.4.3.3 Reinforcement Learning:

In reinforcement learning, an agent learns to make sequential decisions in an environment to maximize a reward signal. It's commonly used in applications like game playing and robotics.

1.4.3.4 Semi-Supervised and Self-Supervised Learning:

These approaches combine elements of supervised and unsupervised learning. In semi-supervised learning, a model is trained with a mix of labeled and unlabeled data. Self-supervised learning often involves creating a supervised task from unlabeled data.

1.4.3.5 summary of top Algorithms used in machine learning:

Machine learning is a vast and evolving field with a wide range of algorithms and techniques. Explaining every machine learning algorithm in detail in a single response would be impractical. However, we provide an overview of some of the most commonly used machine learning algorithms, categorized by their primary purpose in Table 1.3. These are just some of the many machine learning algorithms used for various tasks. The choice of algorithm depends on the specific problem we are trying to solve and the characteristics of the data.

Machine Learning Type	**explanation**	**Common Algorithms**
Supervised Learning	Supervised learning involves training a model on labeled data, where the algorithm learns to map input data to corresponding output labels.	- Linear Regression
		- Logistic Regression
		- Decision Trees
		- Random Forest
		- Support Vector Machines (SVM)
		- K-Nearest Neighbors (K-NN)
		- Naive Bayes
Unsupervised Learning	Unsupervised learning involves working with data that lacks labeled outputs. The goal is to discover patterns or structure within the data.	- K-Means Clustering
		- Hierarchical Clustering
		- Principal Component Analysis (PCA)
		-t-Distributed Stochastic Neighbor Embedding (t-SNE)
		- Apriori Algorithm
Semi-Supervised Learning	These algorithms combine elements of both supervised and unsupervised learning and are often used when there is limited labeled data available.	- Self-training
		- Label Propagation
Reinforcement Learning	Used in scenarios where an agent interacts with an environment and learns to make a sequence of decisions to maximize a reward signal.	- Q-Learning
		- Deep Q Networks (DQN)
		- Policy Gradient Methods
Neural Networks	Neural networks are used for deep learning, and they include various architectures	- Feedforward Neural Networks (FNN)
		- Convolutional Neural Networks (CNN)
		- Recurrent Neural Networks (RNN)
		- Long Short-Term Memory (LSTM) Gated Recurrent Unit (GRU)
		- Transformers

Tableau 1.3: Common Machine Learning Algorithms

1.4.4 The Limitations of Traditional Approaches

Older machine learning methods, while still valuable in many contexts, have some limitations that deep learning methods aim to address. Here are some of the key issues associated with older methods:

- Feature Engineering: Many traditional machine learning algorithms require manual feature engineering, where domain experts need to carefully design and select relevant features from the data. This process can be time-consuming and may not capture complex patterns in high-dimensional data.
- Scalability: Older methods often struggle to scale to large datasets. They may not efficiently handle the vast amount of data generated in today's digital age, which is a problem when working with big data.
- Limited Representation: Traditional machine learning models, such as linear regression and decision trees, have limited capacity to represent complex, non-linear relationships in data. Deep learning models, with their deep neural networks, can learn hierarchical representations of data, allowing them to capture intricate patterns.
- Generalization: Older methods might not generalize well to unseen data, leading to overfitting or underfitting. Deep learning models can, in many cases, generalize better, thanks to their ability to learn hierarchical abstractions.
- Interpretability: Traditional machine learning algorithms are often more interpretable, which can be advantageous in domains where model transparency and explainability are crucial.

It's important to note that the choice between older machine learning methods and deep learning methods depends on the specific problem, data, and resources available. Traditional machine learning methods can still be very effective for many tasks, especially when interpretability and computational resources are limiting factors. Deep learning is a powerful tool but!.

1.4.5 Why deeplearning methods are born

Deep learning methods have emerged and gained prominence for several reasons:

(a) **Availability of Large Datasets:** The advent of the internet and digital technology has led to the generation of vast amounts of data. Deep learning algorithms require a large amount of data to learn from, and the availability of big data made it feasible to train deep neural networks effectively.

(b) **Increased Computational Power:** Deep learning relies on neural networks with many layers (deep architectures). The availability of powerful graphics processing units (GPUs)

and distributed computing frameworks (like TensorFlow and PyTorch) has made it practical to train deep models efficiently.

(c) **Advances in Algorithms:** The development of novel algorithms, such as backpropagation, convolutional neural networks (CNNs), recurrent neural networks (RNNs), and deep reinforcement learning, has enabled deep learning models to handle a wide range of tasks, from image and speech recognition to natural language processing.

(d) **Transfer Learning:** Deep learning has been made more accessible through transfer learning techniques. Pre-trained models like BERT, GPT, and ResNet, enable practitioners to fine-tune models for specific tasks, reducing the amount of data and computational resources required.

(e) **Research Progress:** Researchers in machine learning and artificial intelligence have made significant contributions to the field of deep learning. The development of new architectures, loss functions, optimization techniques, and regularization methods has improved the performance of deep neural networks.

(f) **Industrial and Research Interest:** Deep learning has found applications in various industries, including healthcare, finance, automotive, and more. This practical applicability has led to substantial investments and research efforts in the field.

(g) **Open Source Community:** Many deep learning frameworks and libraries are open source, making it easy for researchers, developers, and practitioners to collaborate and build upon each other's work. This has accelerated the development of deep learning methods.

(h) **Success in Benchmark Tasks:** Deep learning has achieved impressive results in benchmark tasks and competitions, such as image classification, machine translation, and reinforcement learning games like Go and Dota 2. These successes have generated interest and enthusiasm for deep learning.

(i) **Neural Network Architecture Evolution:** Over time, neural network architectures have evolved to become more efficient and effective. For example, the development of convolutional neural networks (CNNs) revolutionized image processing tasks, while recurrent neural networks (RNNs) have excelled in sequential data analysis.

So we can say in summary, deep learning methods have emerged due to the convergence of factors, including data availability, computational resources, algorithmic advances, research progress, and successful applications. This has positioned deep learning as a dominant paradigm in the field of artificial intelligence and machine learning.

1.4.6 Comparison of traditional computer vision methods and deep learning approaches.

Traditional computer vision methods and deep learning approaches represent two distinct paradigms for addressing computer vision tasks. Here's a comparison of these two approaches based on various aspects in Table 1.4:

Aspect	Human Vision	Machine Vision (Traditional)	Computer Vision (Deep Learning)
Biological vs. Artificial Neural Networks	Relies on the complex human brain and biological neurons	Utilizes pre-defined algorithms and image processing techniques	Utilizes artificial neural networks inspired by the human brain's structure.
Learning Mechanisms	Acquired through years of exposure and learning	Utilizes pre-defined algorithms and parameter settings	Requires training data and deep learning algorithms for learning and pattern recognition.
Subjectivity vs. Objectivity	Inherently subjective, influenced by individual experiences and emotions	Objective, particularly in traditional systems for specific tasks	Designed for objectivity, providing consistent and data-driven assessments.
Processing Speed and Data Volume	Quick adaptation but may not handle large data volumes efficiently	Capable of processing data but not as rapidly or efficiently as deep learning	Processes large datasets rapidly with high accuracy, suitable for real-time tasks.
Adaptability and Generalization	Highly adaptable and capable of generalization	Limited adaptability and generalization in traditional systems	Adapts and generalizes to some extent but may require retraining for novel situations or objects.

Tableau 1.4: Comparison of Human Vision, Machine Vision (Traditional), and Computer Vision (Deep Learning)

We notice from the Table of Comparison 1.4, that traditional computer vision methods offer transparency and are well-suited to specific, well-defined tasks with limited variations. And deep learning approaches, on the other hand, excel in handling complex and diverse data, requiring less manual feature engineering but may be computationally intensive and have challenges related to interpretability. The choice between the two approaches depends on the specific requirements and constraints of a given computer vision task.

1.5 Deep learning

Deep learning is a subset of machine learning that uses neural networks with multiple layers to learn and extract hierarchical representations of data. These networks are trained on large datasets and can autonomously learn complex features and patterns without explicit instructions. Deep learning has been successfully applied in various fields, including image and speech recognition, natural language processing, and computer vision. [24, 25]

1.5.1 Deep Learning Challenges:

While deep learning techniques have demonstrated remarkable prowess in solving complex tasks through multiple layers and high levels of abstraction, it is widely acknowledged that the accuracy, sharpness, responsiveness, and precision of deep learning systems can match or even exceed those of human experts. However, to fully harness the potential of this technology and achieve continued success, it must confront numerous challenges. Therefore, presented below is a compilation of the challenges that deep learning must address:

- Data Quality and Quantity: Acquiring and maintaining large, high-quality datasets.
- Overfitting: Preventing models from fitting training data too closely.
- Complexity and Interpretability: Understanding and explaining model decisions.
- Training Time and Computational Resources: Demanding hardware and time for model training.
- Hyperparameter Tuning: Finding optimal settings for model parameters.
- Vanishing and Exploding Gradients: Managing gradient issues in deep networks.
- Generalization: Ensuring models perform well on unseen data.
- Ethical and Bias Concerns: Addressing bias and ethical implications in model predictions.
- Security and Adversarial Attacks: Safeguarding models against attacks.
- Limited Data Efficiency: Improving model performance with limited data.
- Model Size and Deployment: Optimizing and deploying large models.
- Sustainability: Reducing the environmental impact of model training.
- Lack of Understanding: Theoretical foundations for deep learning are incomplete.
- Scalability: Extending deep learning to more complex problems.
- Continual Learning: Enabling models to learn from new data while retaining past knowledge.

A good survey was conducted in 2014 [26]. In this paper, the authors explained details on how DL can deal with different criteria, including volume, velocity, variety, and veracity of the big data problem. And in [27] they provided an in-depth understanding of deep learning techniques, including the challenges and solutions.

1.5.2 Terminologies of DL:

Artificial Neuron vs Biological Neuron :

Artificial Neurons (ANs) and Biological Neurons (BNs) (both in Figure 1.10) share many basic components, but there are significant differences in their complexity, flexibility, and adaptability. BNs are highly complex and adaptable systems that can process information in parallel, and their plasticity allows them to learn and adapt over time. In contrast, ANs are simpler systems that are designed to perform specific tasks, and their connections are usually fixed, with the network architecture determined by the designer.

An artificial neuron also known as a perceptron is a computational unit of the neural network that takes an input and produces an output, serving as a mathematical representation of a biological neuron. It operates by applying a mathematical function, known as an activation function, to the input, yielding the output.

Each artificial neuron has the following main functions:

- Takes inputs from the input layer
- Weighs them separately and sums them up
- Pass this sum through a nonlinear function to produce output.

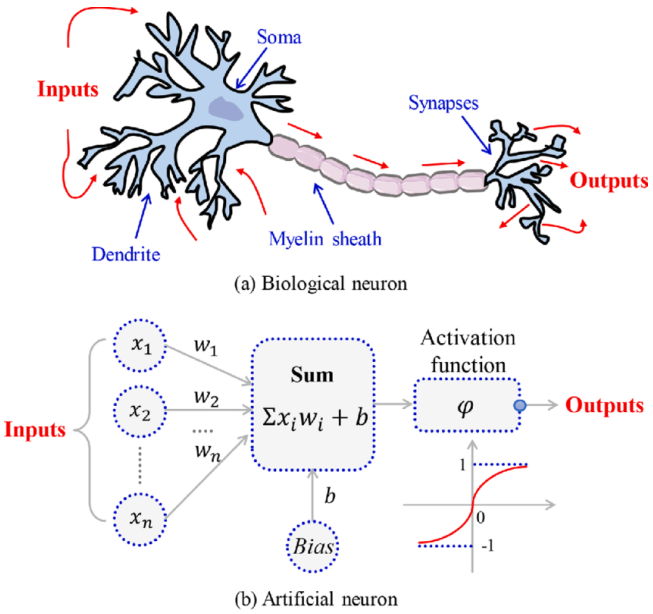


Figure 1.10: Comparison between biological neuron and artificial neuron. [2]

we can now draw comparisons between both as follows 1.11:

Biological Neuron	Artificial Neuron
Dendrites	Input
Cell Nucleus(Soma)	Node
Axon	Output
Synapse	Interconnections

Figure 1.11: Similarities between biological neuron and artificial neuron.

Artificial neural networks :

An artificial neural network (ANN) (figure 1.10) is a computational model inspired by the structure and function of biological neural networks in the human brain. It consists of interconnected nodes, called neurons, organized into layers. ANNs are designed to process and learn from data by adjusting the strengths of connections (weights) between neurons to make predictions, recognize patterns, or solve various tasks, such as classification, regression, and decision-making. They are a fundamental component of deep learning and are used in a wide range of applications.

Artificial neural networks faced significant challenges in the past, which hindered their progress and adoption. ANNs were heavily handicapped by:

1. Lack of Large, Labeled Datasets: To train deep neural networks effectively, you need a large amount of labeled data. During the 1980s,1990s and 2000s, obtaining such datasets was a major challenge. The availability of large labeled datasets, like ImageNet, played a crucial role in the resurgence of deep learning in the 2010s.

2. Computational Power: Training deep neural networks can be computationally intensive. The lack of powerful hardware made it impractical to train large networks with many layers. Advances in hardware, including GPUs and specialized accelerators, have greatly improved the speed of training deep networks.

3. Inadequate Activation Functions: The choice of activation functions is critical in neural networks. In the past, some suboptimal activation functions were used, which limited the network's ability to learn complex patterns. The development of better activation functions, such as rectified linear units (ReLU), significantly improved training.

4. Weight Initialization: Proper weight initialization is essential for training deep networks. Poor weight initialization can lead to vanishing or exploding gradients, making training difficult. Modern techniques for weight initialization, like He initialization, have helped address this issue.

All these challenges, contributed to the limitations of training deep neural networks in the past.

However, over the years, researchers have developed solutions to these problems, leading to the resurgence of deep learning and the successful training of deep neural networks with many hidden layers.

Deep learning has since become a powerful tool in various fields, including computer vision, natural language processing, and reinforcement learning.

1.5.3 Subdomain of DL

- **Convolutional Neural Networks (CNNs) [28]:** they are a specific type of artificial neural network crafted to handle data structured in a grid-like topology, such as images, speech signals, and time-series data. Comprising multiple layers of interconnected neurons, they execute convolutions, pooling operations, and non-linear transformations on the input data. This process enables the network to discern and extract high-level features and patterns from raw inputs effectively. CNNs have demonstrated exceptional proficiency across a spectrum of tasks, including image and speech recognition, natural language processing, and autonomous driving, among others.

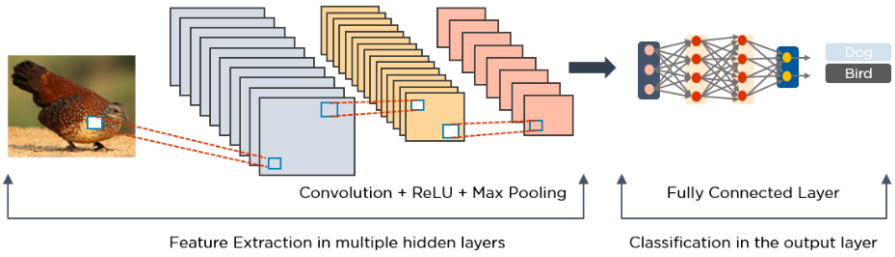


Figure 1.12: The architecture of Convolutional Neural Networks

- **Recurrent Neural Networks (RNNs) [29]:** are a specific type of ANNs designed to analyze sequential data by integrating feedback loops within the network structure. This design enables RNNs to retain a state or memory of previous inputs and utilize it to generate predictions or decisions based on current input. Their capacity to manage sequences of variable lengths and capture temporal dependencies renders RNNs well-suited for tasks such as language modeling, speech recognition, and natural language processing.

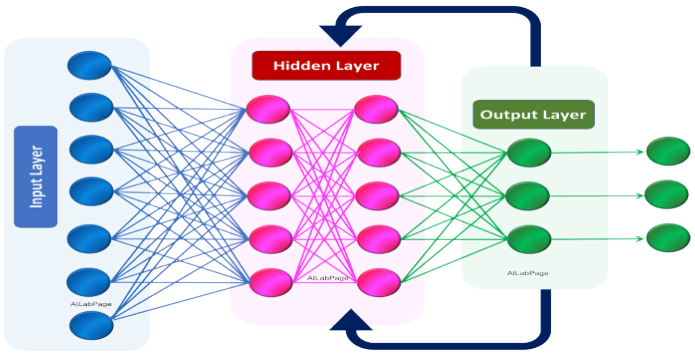


Figure 1.13: The architecture of Recurrent Neural Networks

- **Generative Adversarial Networks (GANs) [30]:** are a category of deep learning models described by two neural networks: a generator and a discriminator. The generator network's purpose is to learn how to generate synthetic data samples resembling real data from a given training set. Conversely, the discriminator network learns to distinguish between the generated data and the real data. Through adversarial training, GANs strive

to create synthetic data of high quality that closely resembles real data, thus facilitating various applications like image and video generation.

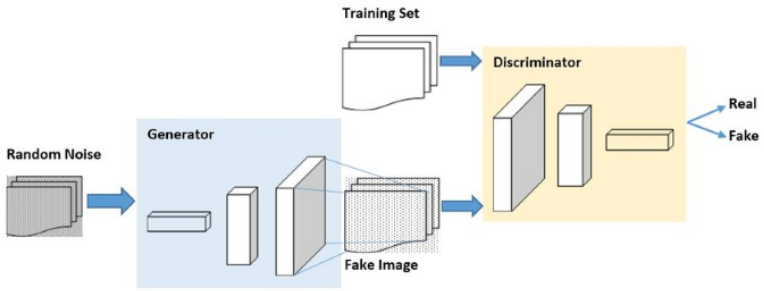


Figure 1.14: The architecture of Generative Adversarial Networks

- **Reinforcement Learning (RL) :** Reinforcement Learning [31] is a branch of machine learning where an agent learns to navigate an environment through trial and error. By taking actions and observing their outcomes, the agent aims to maximize a cumulative reward signal provided by the environment. RL draws inspiration from human learning processes and has found success in various fields like robotics, gaming, and autonomous vehicles, where adaptive decision-making is crucial.

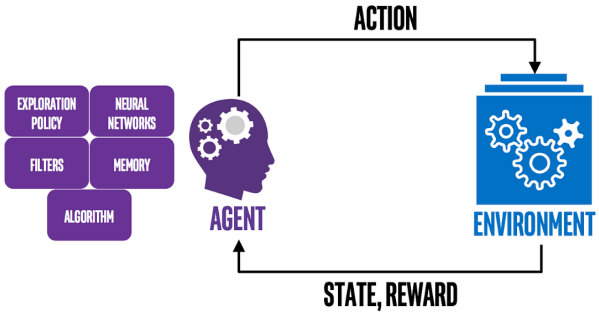


Figure 1.15: The architecture of Reinforcement Learning

- **Transfer Learning [3]:** Transfer learning is a methodology within machine learning that leverages pre-existing models trained on one task to be reused on a new related task. By repurposing learned knowledge, it streamlines the learning process, diminishes the need for extensive training data, and accelerates training. This technique is instrumental in various domains, facilitating the adaptation and fine-tuning of models to tackle new challenges efficiently.

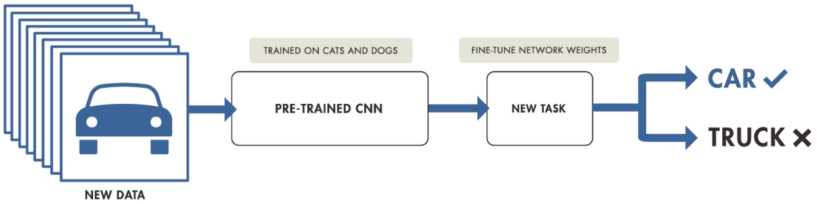


Figure 1.16: The architecture of Transfer Learning. [3]

- **Attention Mechanisms [32]:** Attention mechanisms are a set of techniques employed within deep learning models to enhance their performance, particularly when dealing with sequential or set-like data. These mechanisms enable models to selectively focus on specific parts of the input data, assigning varying degrees of importance to different components. This selective attention allows models to effectively handle long-term dependencies, accommodate variable-length input sequences, and mitigate overfitting. At the core of attention mechanisms is the computation of attention weights, which determine the contribution of each input element to the output. By dynamically adjusting these weights during the training process, attention mechanisms allow models to intelligently weigh and aggregate information, resulting in more accurate and contextually relevant predictions. Overall, attention mechanisms play a crucial role in improving the interpretability and performance of deep learning models, particularly in tasks involving sequential or set-like data, such as natural language processing, speech recognition, and image captioning.

- **Autoencoders for Unsupervised Learning and Feature Extraction [33]:** Autoencoders are a class of neural networks that excel in unsupervised learning and feature extraction tasks. They consist of two primary components: an encoder and a decoder. The encoder compresses the input data into a latent space representation, while the decoder aims to reconstruct the original input from this compressed representation. This process occurs through a bottleneck layer within the network. The key objective of an autoencoder is to minimize the reconstruction error between the input and the output. By doing so, the network is encouraged to learn meaningful representations of the data in the latent space. These learned representations capture important features of the input data, making autoencoders invaluable for tasks such as dimensionality reduction, anomaly detection, and data denoising.

1.5.4 Advanced Visual Techniques in Deep Learning

In the field of deep learning within computer vision, several fundamental tasks are pivotal for analyzing and understanding images or data. These tasks encompass classification, detection, and segmentation, each serving distinct purposes in extracting meaningful insights from visual data, involving analyzing and understanding images.

- **Object detection:** Object detection involves identifying and localizing the presence of objects within an image or video, typically by drawing bounding boxes around them. This process entails not only determining the object's class but also pinpointing its precise location in the visual data. Object detection algorithms leverage deep neural networks for feature extraction and classification, significantly enhancing accuracy and efficiency in various domains such as surveillance, robotics, medical imaging and autonomous vehicles [34].

- **Classification :** Classification revolves around assigning labels or categories to inputs based on their features. In the context of image classification, deep learning models are trained on extensive datasets containing thousands of labeled images to learn patterns and recognize different objects or concepts, accurately assigning corresponding labels. This process harnesses and use neural networks alongside techniques like convolutional layers, pooling, and activation functions to extract and transform input features for classification [35].

- **Segmentation :** Image segmentation entails partitioning an image into multiple labeled convex called segments or regions, with each segment representing a distinct object or part of the image. This task involves labeling every pixel within the image to indicate its corresponding object or region. Deep learning approaches, particularly convolutional neural networks (CNNs) (The most well-known are UNet and Mask R-CNN), they have been instrumental in image segmentation tasks, showing promising outcomes across diverse applications such as medical imaging, autonomous driving, and remote sensing [36].

When we compare them we can notice that while classification assigns labels to entire images and detection localizes and labels objects within images, segmentation goes further by labeling every pixel, offering granular insights into image composition and content (Figure 1.17).

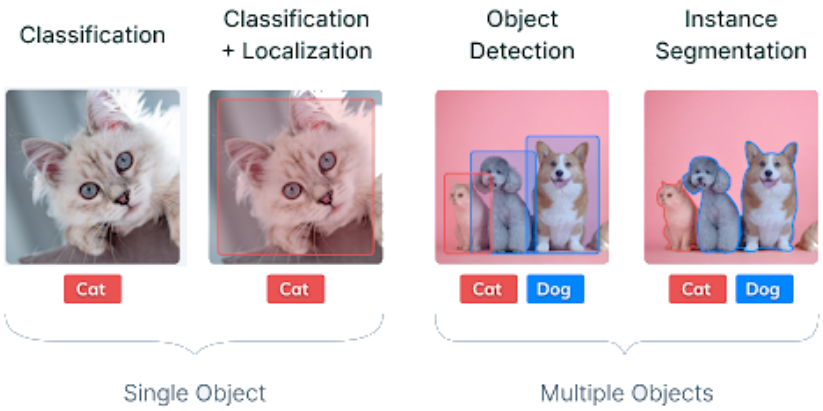


Figure 1.17: classification Vs detection Vs segmentation

1.6 Real-Time Object Detection

Real-time object detection is a computer vision challenge that entails identifying and localizing objects in live video streams with high accuracy and rapid inference. By integrating object detection and tracking methods, algorithms can precisely detect and track objects in real time. These algorithms utilize feature extraction, object proposal generation, and classification to identify and localize objects of interest. The domain of real-time object detection finds varied

applications, including autonomous driving, surveillance, robotics, and the development of AI-driven solutions for individuals with visual impairments.

The impact of real-time object detection using deep learning on AI-based solutions for individuals with visual impairments is substantial. This technology enables the creation of visual assistance systems that empower individuals with visual impairments to navigate their surroundings more independently and securely. For instance, object detection can assist in recognizing and locating objects such as stairs, doors, obstacles, as well as identifying faces and emotions.

Moreover, real-time object detection facilitates the advancement of assistive technologies for the visually impaired, including wearable devices capable of detecting objects and providing auditory or haptic feedback to the user. These solutions have the potential to significantly improve the quality of life for individuals with visual impairments by enhancing their independence and mobility.

Several features can influence real-time object detection:

- **Model architecture:** The selection of model architecture can significantly impact the speed and accuracy of real-time object detection. For example, models like YOLO (You Only Look Once) are optimized for real-time performance and can process images in real time, while others like R-CNN offer higher accuracy at the cost of speed.
- **Image resolution:** The resolution of the input image also affects the speed and accuracy of object detection. Generally, lower-resolution images can be processed faster but may result in reduced accuracy.
- **Hardware:** The choice of hardware, such as the GPU type, can significantly affect the speed of real-time object detection.
- **Preprocessing techniques:** Techniques like image cropping and scaling can improve the speed and accuracy of object detection.
- **Object density and complexity:** The number and complexity of objects in the image can also impact the speed and accuracy of real-time object detection. Images with numerous objects or objects with intricate shapes may require more processing time and could affect real-time performance.

In conclusion, achieving the optimal balance between ***accuracy*** and ***speed*** in real-time object detection necessitates considering a combination of these factors.

1.6.1 The key components of a real-time object detection pipeline.

A real-time object detection pipeline is a series of interconnected processes designed to identify and locate objects within images or video frames in real-time. This pipeline is typically used in applications such as autonomous vehicles, surveillance systems, robotics, and more. The key components of a real-time object detection pipeline include:

(a) ***Input Source:*** The pipeline begins with an input source, which can be a live video feed, a pre-recorded video, or a sequence of images. This source provides the data on which object detection will be performed.

(b) ***Image Preprocessing:*** Image preprocessing is an essential step that may involve resizing, normalization, and enhancement of the input images or frames. Preprocessing can improve the efficiency and accuracy of object detection by ensuring that the data is in the right format and quality.

(c) ***Feature Extraction:*** In traditional computer vision pipelines, this step involves extracting relevant features from the images, such as edges, corners, or color histograms. In deep learning-based pipelines, feature extraction is often embedded within the neural network layers, allowing the model to automatically learn and extract meaningful features.

(d) ***Object Detection Model:*** The heart of the pipeline is the object detection model. This is where deep learning models, like Faster R-CNN, YOLO (You Only Look Once), or SSD (Single Shot MultiBox Detector), come into play. The model takes the preprocessed image as input and outputs the detected objects along with their bounding boxes and class labels.

(e) ***Post-Processing:*** After the model's predictions, post-processing is applied to refine and filter the results. Common post-processing techniques include non-maximum suppression (NMS) to remove duplicate or overlapping bounding boxes, setting confidence thresholds, and filtering out small or irrelevant detection.

(f) ***Visualization:*** To provide a visual representation of the detected objects, the pipeline may include components for drawing bounding boxes around the objects and labeling them with their corresponding class names.

(g) ***Tracking (Optional):*** In applications where real-time object tracking is required, an additional tracking module can be integrated into the pipeline. This component helps maintain the continuity of object identities across frames, allowing for tracking moving objects over time.

(h) ***Output:*** The final output typically includes a video feed or a series of images with bounding boxes drawn around detected objects, along with their corresponding class labels. In some applications, this output may also be used for decision-making or further analysis.

(i) ***Feedback Loop (Optional):*** In interactive systems like robotics or autonomous vehicles, a feedback loop can be incorporated to adapt the object detection pipeline based on the system's real-time performance and user requirements. This may involve retraining the model, adjusting parameters, or altering the processing flow dynamically.

(j) ***Real-Time Execution:*** The entire pipeline is designed to operate in real-time, which means that it must process images or frames within a defined time frame (e.g., 30 frames per second for live video) to provide timely and responsive results.

These components work together to enable real-time object detection, making it suitable for a wide range of applications where immediate detection and tracking of objects in a dynamic environment are crucial. The choice of specific algorithms, models, and parameters may vary depending on the application and hardware constraints.

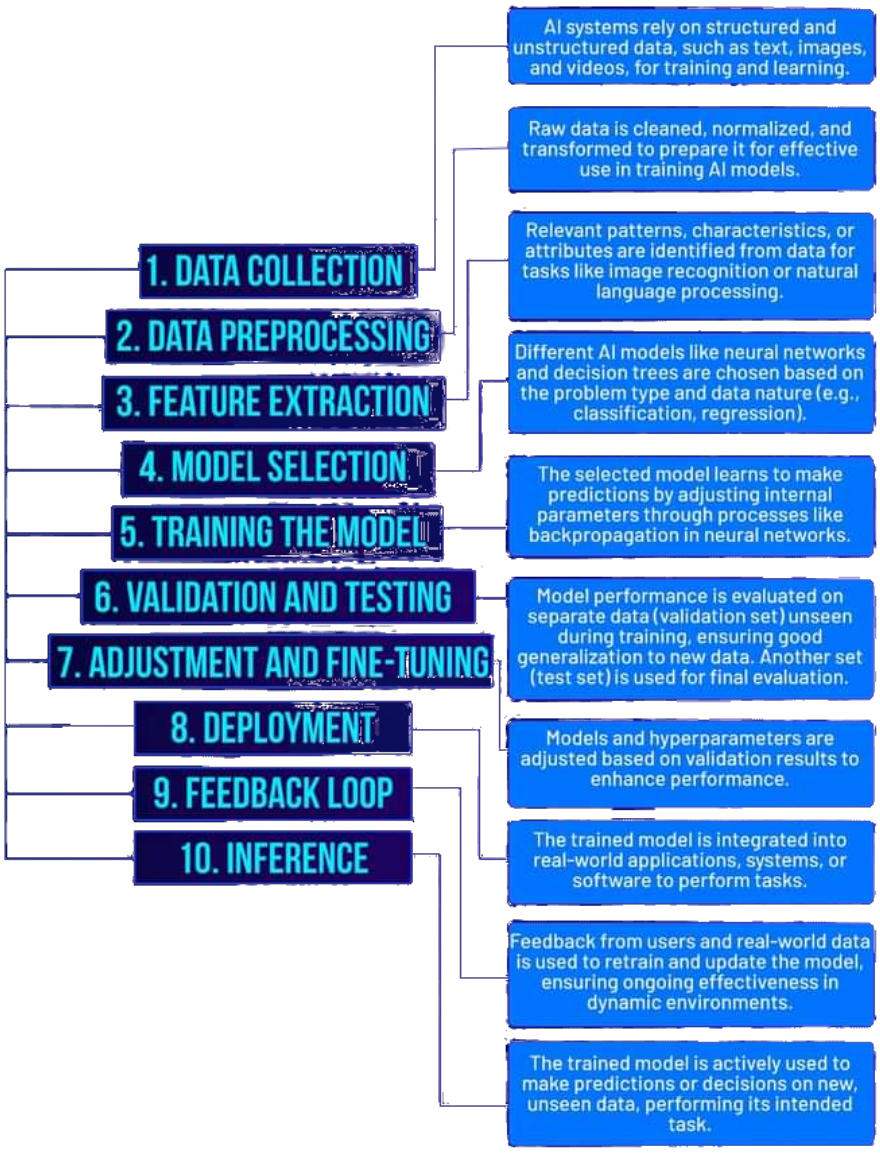


Figure 1.18: object detection pipeline

1.6.2 inspiration from human objects detection pipeline

Detecting and recognizing objects in computer vision often draws inspiration from how the human visual system works. While the mechanisms are not identical, there are parallel components in both human and computer vision pipelines. In human object detection, key components include:

- ***Sensory Input (Vision):*** In human vision, the process starts with the eyes, which act as sensors capturing visual information from the surrounding environment. The parallel component in computer vision is the input source, which can be live camera feeds or pre-recorded images and videos.
- ***Preprocessing (Low-Level Processing):*** Just as computers perform preprocessing on images, the human visual system conducts low-level processing. This includes activities like adjusting for lighting conditions, removing noise, and enhancing the image to improve visual perception.
- ***Feature Extraction (Early Visual Processing):*** In computer vision, feature extraction involves identifying important visual cues. Similarly, in the human visual system, feature extraction occurs in the early visual processing stages, where the brain identifies basic visual elements like edges, shapes, and colors.
- ***Object Recognition (Object Detection):*** Object recognition in computer vision involves identifying and localizing objects within images or video frames. In the human visual system, this corresponds to recognizing objects and understanding their attributes (e.g., identifying a cat and knowing it's furry, has four legs, and a tail).
- ***Higher-Level Processing (Cognition):*** After detecting and recognizing objects, both systems engage in higher-level processing. In computer vision, this can involve analyzing the detected objects' attributes or behaviors (e.g., recognizing a moving vehicle and predicting its trajectory). In the human visual system, this is where cognition and reasoning come into play, allowing us to understand objects in context and make decisions based on what we see.
- ***Feedback Loop (Learning and Adaptation):*** Both computer vision systems and human vision involve a feedback loop for learning and adaptation. In computer vision, this may include retraining machine learning models or adjusting parameters to improve detection accuracy. In the human visual system, this corresponds to the learning and adaptation that occurs over time as individuals gain more experience and expertise.
- ***Real-Time Processing:*** In both systems, real-time processing is critical. Whether it's in real-time video analysis for computer vision applications or the continuous perception and interpretation of the visual world by humans, timely processing is essential for making immediate decisions and reactions.

It's important to note that while there are parallel components, the mechanisms by which computer vision and the human visual system achieve these tasks are fundamentally different. Human vision is a highly complex and nuanced process that involves the integration of multiple sensory modalities, cognitive functions, and learning over time. Computer vision, on the other hand, relies on algorithms and mathematical models to replicate certain aspects of visual perception.

1.7 Object Detection

Object Detection is a computer vision task in which the goal is to detect and locate objects of interest in an image or video. The task involves identifying the position and boundaries of objects in an image, and classifying the objects into different categories, like in figure 1.19. It forms a crucial part of vision recognition, alongside image classification and Instance segmentation.

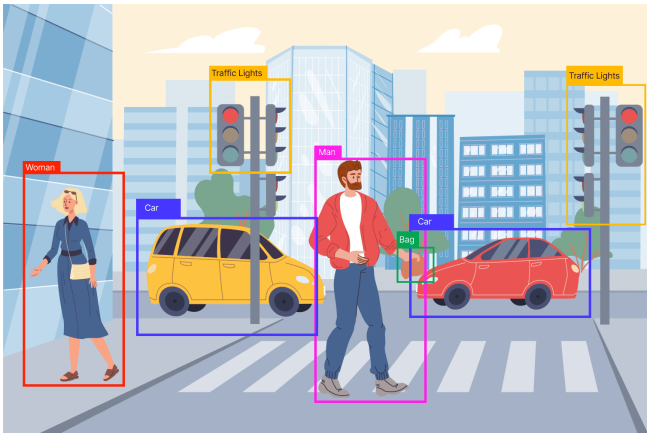


Figure 1.19: Object Detection

The key characteristics of object detection include:

(a) Object Localization: Object detection not only recognizes the presence of objects in an image but also provides information about their spatial location. This typically involves drawing bounding boxes around the objects to indicate their position.

(b) Multiple Object Recognition: Object detection can identify and locate multiple objects of different classes within a single image. This distinguishes it from simple image classification, which assigns a single label to an entire image.

(c) Object Classification: In addition to localization, object detection also involves classifying each object into predefined categories or classes, such as cars, pedestrians, or animals.

Object detection can be approached using various algorithms and models, including traditional computer vision techniques like Haar cascades and Histogram of Oriented Gradients (HOG) [37], as well as deep learning methods such as Convolutional Neural Networks (CNNs) and more advanced architectures like Region-based CNNs (R-CNN) [38], Fast R-CNN [39], Faster R-CNN [40], YOLO (You Only Look Once) [7], and SSD (Single Shot MultiBox Detector) [41]. These deep-learning models have significantly improved the accuracy and efficiency of object detection tasks, making them particularly popular in recent years. They are capable of real-time object detection, making them suitable for applications where speed and accuracy are critical.

1.7.1 Utilizing Object Detection for Personal Problem-Solving

An important question we have to deal with: "How do we use Object Detection to solve my own problem?" Object detection can serve as a solution to a wide range of questions, falling into the following broad categories:

- Is an object present in my Image or not?
- Where is an object in the image?
- How many objects are there in an image?
- What are the different types of objects in the Image?
- What is the size of an object?
- How are different objects interacting with each other?
- Where is an object with respect to time (Tracking an Object)?

1.7.2 Challenges in Object Detection

Object detection is a challenging task in computer vision due to various complexities and factors. Some of the key challenges in object detection include:

- Scale Variability: Objects in images can appear at different scales, and it can be challenging to detect them accurately regardless of their size. Some objects may be tiny, while others can be quite large.
- Occlusion: Objects are often partially or fully occluded by other objects in the scene. Handling occluded objects is a challenging aspect of object detection.
- Viewpoint Variability: Objects may appear from different angles and viewpoints. An effective object detection system should be able to recognize objects regardless of their orientation.
- Illumination Changes: Variations in lighting conditions, such as shadows, highlights, and different times of day, can make object detection more difficult.
- Object Deformation: Objects can deform or change shape, making it challenging to match the object with predefined templates or models.
- Limited Training Data: Training object detection models typically requires a large dataset of annotated images. In some cases, acquiring such data can be expensive or time-consuming.

- Generalization: Ensuring that an object detection model can generalize well to new, unseen data is a significant challenge. Over-fitting (performing well on training data but poorly on new data) is a common issue.
- Class Imbalance: In some scenarios, certain object classes may be rare, leading to class imbalance in the training data. This can affect the model's ability to recognize less common objects.
- Adaptation to Different Domains: Object detection models trained on one dataset or environment may not perform well in a different domain or under different conditions. Domain adaptation is a challenging problem.
- Robustness to Noise: Images may contain noise, artifacts, or distortions that can affect the performance of object detection models.
- Cluttered backgrounds: Objects are often found in complex and cluttered scenes, which can make it challenging to distinguish the object of interest from the surrounding environment.
- Limited computational resources: In resource-constrained environments, such as embedded systems or mobile devices, there may be limitations on the computational power available for object detection algorithms.
- Multi-object detection: Detecting multiple objects in the same scene, especially when objects are close together or overlapping, presents challenges in accurately identifying and localizing each object.

Addressing these challenges often requires the development of advanced algorithms, the use of large and diverse datasets, and fine-tuning models to handle specific scenarios. Researchers and engineers continuously work to improve object detection methods to make them more robust and reliable across a wide range of applications.

1.7.3 Object Detection Models

Object detection in computer vision is commonly divided into two categories: single-stage and two-stage detectors.

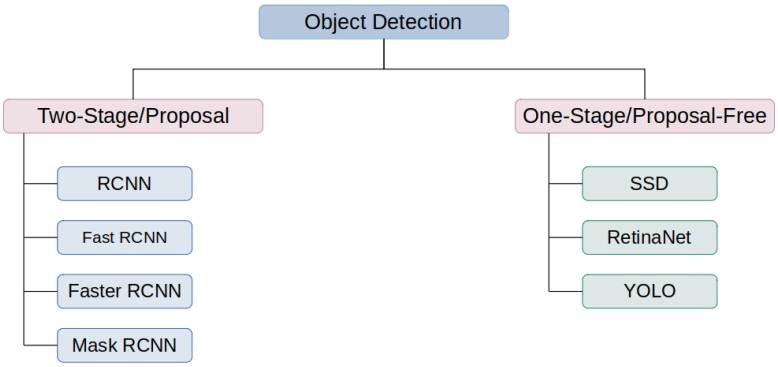


Figure 1.20: Object Detection categories

(a) **Two-stage object detectors:** These detectors work in two distinct steps. First, they identify a set of potential regions or proposals within the image that could contain objects. Then, these regions are refined and classified to determine the final object detection results. This method is known for its accuracy in precisely locating objects and is typically preferred for tasks where accuracy is crucial.

(b) **Single-stage object detectors:** In contrast, single-stage detectors aim to detect objects directly in a single pass through the network. They predict both the bounding boxes and class labels of objects without the need for a separate proposal generation step. This approach prioritizes speed and efficiency, making it suitable for real-time applications. Single-stage detectors often utilize predefined anchor boxes or default priors to localize and classify objects efficiently.

Two-stage detectors focus on accuracy by employing a two-step process, while single-stage detectors prioritize speed and efficiency by detecting objects directly in a single pass through the network.

1.7.4 Choosing the Best Object Detection Model

Selecting the ideal model for an object detection system involves several critical factors. Here's a breakdown of key considerations to guide your decision-making process:

(a) **Accuracy Requirements:** Define the required accuracy level for your application. Some models prioritize accuracy over speed, while others strike a balance between the two. Evaluate the trade-off between accuracy and inference time based on your project's needs.

(b) **Speed and Latency:** For applications requiring real-time or near-real-time performance, prioritize models with fast inference times. Single-shot models like YOLO and SSD generally outperform two-stage models like Faster R-CNN in speed. Consider available hardware and computational resources when assessing speed requirements.

(c) **Model Size and Resource Constraints:** Consider the model's size and the computational resources available. Larger models with more parameters may offer higher accuracy but demand more memory and processing power. Opt for efficient models like MobileNet [42] or EfficientDet [43] if resources are limited.

(d) **Dataset and Domain Relevance:** Evaluate the model's relevance to your dataset and application domain. Models pretrained on large-scale datasets such as COCO or ImageNet may offer better generalization and performance. Also, ensure the model has been trained on objects and classes similar to those in your dataset.

(e) **Implementation and Deployment:** Check for pre-trained models, frameworks, and libraries supporting your chosen model. Well-documented implementations and resources simplify development and integration into your system.

(f) **State-of-the-Art Advancements:** Stay updated on the latest research and advancements in object detection. Newer models often introduce improvements in accuracy, efficiency, or additional features. Explore recent publications and benchmark results to identify models that excel in your specific use case.

(g) **Evaluation and Comparison:** Assess multiple models using metrics like mAP (mean Average Precision) and F1 score on your dataset or benchmark. Experiment with different configurations and hyper parameters to determine the best-performing model for your task.

Ultimately, the optimal object detection model depends on the specific requirements, constraints, and priorities of your application. Experimentation and fine-tuning may be necessary to strike the right balance between accuracy, speed, and resource utilization.

1.7.5 The best metrics

The choice of metrics for evaluating any proposed solution for object detection, will depend on the specific objectives of the system. However, some commonly used metrics that can be used to evaluate the performance of such a system include:

(a) **Object detection accuracy:** This metric measures how accurately the system is able to detect objects. It can be evaluated using measures such as precision, recall, and F1 score, which provide a quantitative measure of the system's ability to detect objects.

(b) **Real-time performance:** This metric measures how quickly the system is able to process images and detect objects in real-time. It can be evaluated using measures such as frames per second (FPS) and processing time per frame.

(c) **Accessibility:** This metric measures how accessible the system is to users. It can be evaluated using measures such as the level of users that the system is able to accommodate, the ease of use of the system, and the availability of alternative input/output methods.

(d) **Cost-effectiveness:** This metric measures the cost-effectiveness of the system in terms of the benefits it provides to users of this kind of solutions. It can be evaluated using measures such as the cost of the hardware and software components used in the system, the cost of maintaining and updating the system, and the overall benefits provided to users.

(e) **User satisfaction:** This metric measures how satisfied the user is with the system's performance and usability. It can be evaluated using measures such as user feedback surveys or user testing sessions.

By evaluating the system using these metrics, it will be possible to identify areas for improvement and optimization, as well as to assess the overall effectiveness and usability of the system.

1.7.6 Object detection approach

Over the last years, there have been significant advancements in object detection approaches, driven largely by developments in deep learning and neural network architectures. One of the most prominent and widely-used object detection approaches in recent years are showen in Figure 1.21 where we can see the usage over time between 2017 and 2023, of the most important Object detection approaches.

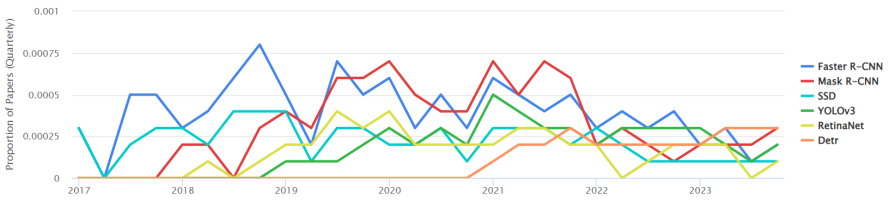


Figure 1.21: Object Detection Models Usage Over Time

1.7.6.1 R-CNN

R-CNN, also known as Regions with CNN Features , proposed by Ross Girshick in 2014 [38], is an object detection model that employs powerful Convolutional Neural Networks (CNNs) to propose regions from the bottom-up, aiming to locate and segment objects within an image. It utilizes a method called selective search to detect various bounding-box object regions, termed "regions of interest", and then proceeds to extract features from each region separately for the purpose of classification. He introduced the concept of region-based convolutional neural networks.

The key modules of the R-CNN model, are:

(a) **Regional Proposal Generation:** This module generates candidate detections that are independent of the object category, providing a set of potential regions of interest for the detector.

(b) **Feature Extraction:** The second module employs a large convolutional neural network to extract fixed-length feature vectors from each region of interest.

(c) **Classification and Localization:** The third module involves using class-specific linear SVMs for both classifying and localizing the objects.

The performance of R-CNN on the PASCAL VOC 2010 dataset and the ILSVRC2013 object detection dataset, demonstrates its effectiveness:

- On the PASCAL VOC 2010 dataset, R-CNN achieves an average precision (mAP) of 53.7%. This performance significantly outperforms alternative approaches in that time, such as utilizing the same region proposals but employing a spatial pyramid and bag-of-visual-words approach, which results in an average precision of 35.1%.
- On the ILSVRC2013 object detection dataset with 200 classes, R-CNN achieves an mAP of 31.4%, representing a significant improvement over the previous best result of 24.3% achieved by OverFeat.

1.7.6.2 Fast R-CNN

Fast R-CNN (lso proposed by Ross Girshick in [39]) is an enhanced version of the R-CNN and SPPNet object detection modelq. It incorporates a Region Proposal Network (RPN) directly into the model for efficient region proposal generation, adopts a unified network architecture, introduces Region of Interest (RoI) pooling for feature extraction, simplifies training with end-to-end training, and achieves faster and more accurate object detection compared to R-CNN. that's why it's called Fast R-CNN,

This approach, named for its notably faster training and testing capabilities, involves processing entire images and a set of object proposals to generate a feature map through multiple convolutional and max pooling layers.

For each object proposal, Fast R-CNN utilizes a region of interest (RoI) pooling layer to extract a fixed-length feature vector from the feature map. This vector then undergoes processing through fully connected layers, leading to two output layers: one producing softmax probability estimates for K object classes (including a "background" class), and the other generating refined bounding box positions with four real-valued numbers for each of the K object classes.

Fast R-CNN achieves impressive results, with a performance of 66.1% on VOC2010 and the best performance on VOC12, achieving an mAP of 65.7% (and 68.4% with additional data).

1.7.6.3 Faster R-CNN

This architecture, introduced by Shaoqing Ren et al. [40], provides a robust solution for object detection by integrating two modules that work together to generate accurate object proposals and efficiently perform object classification. These modules are as follows:

(a) The first module consists of a deep convolutional network that produces a feature map through convolutional operations. This feature map serves as input to the Region Proposal Network (RPN) module, which generates a set of rectangular object proposals along

with their corresponding precision scores. The RPN is designed to handle feature maps of varying sizes.

(b) The second module is the Fast R-CNN detector, which utilizes the proposed regions based on the principles of the Fast R-CNN architecture.

When the model is evaluated on the MSCOCO dataset (specifically, the COCO validation set), it achieves a mean average precision (mAP) of 41.5% when considering an intersection over union (IoU) threshold of 0.5. Similarly, on the COCO test-dev set, the model attains an mAP of 42.7% at an IoU threshold of 0.5, and an mAP of 21.9% within the IoU range of 0.5 to 0.95. These evaluation results on the MSCOCO dataset exhibit competitive performance in terms of mAP scores, and highlight the model's performance.

1.7.6.4 Mask R-CNN

Kaiming He et al. [44] proposed an extended architecture called Mask R-CNN, for object detection and instance segmentation. It includes a backbone network, a Region Proposal Network (RPN) for object detection, and a segmentation branch for pixel-wise object delineation. This branch operates in parallel with the existing branch responsible for bounding box recognition

It uses RoI Align to preserve pixel-level details, predicts masks for each proposed region, and is trained end-to-end with a multi-task loss function. During inference, it generates candidate object regions, extracts features, and predicts class labels, bounding boxes, and segmentation masks. This mask branch employs a small Fully Convolutional Network (FCN) applied to each Region of Interest (RoI), enabling pixel-level segmentation mask prediction. This pixel-wise alignment is a crucial addition to the Fast/Faster R-CNN frameworks. Mask R-CNN is effective for tasks requiring precise object segmentation, such as medical imaging and robotics.

When evaluated on the MS COCO dataset, including fine-tuning using the ResNet-50-FPN backbone, the model achieves a validation (mAP) of 36.4%.

1.7.6.5 SSD (Single Shot MultiBox Detector)

The SSD (Single Shot MultiBox Detector) method, outlined in [41], employs a feed-forward convolutional network to produce a fixed set of bounding boxes and corresponding scores, indicating the presence of object instances across different classes. These detections undergo non-maximum suppression to refine the final results. Initially, the network utilizes standard layers, like those from VGG-16, typically used for high-quality image classification, with the classification layers removed (referred to as the base network).

Furthermore, the network incorporates an auxiliary structure to enhance key detection features, including multi-scale feature maps, convolutional predictors, and default boxes with predefined aspect ratios. The SSD512 model, trained on the COCO trainval35k dataset and further refined using Pascal VOC 2007+2012, achieves a (mAP) of 81.6%. In contrast, the SSD300 model, trained solely on the VOC 2007 dataset, achieves a lower mAP of 68.0%.

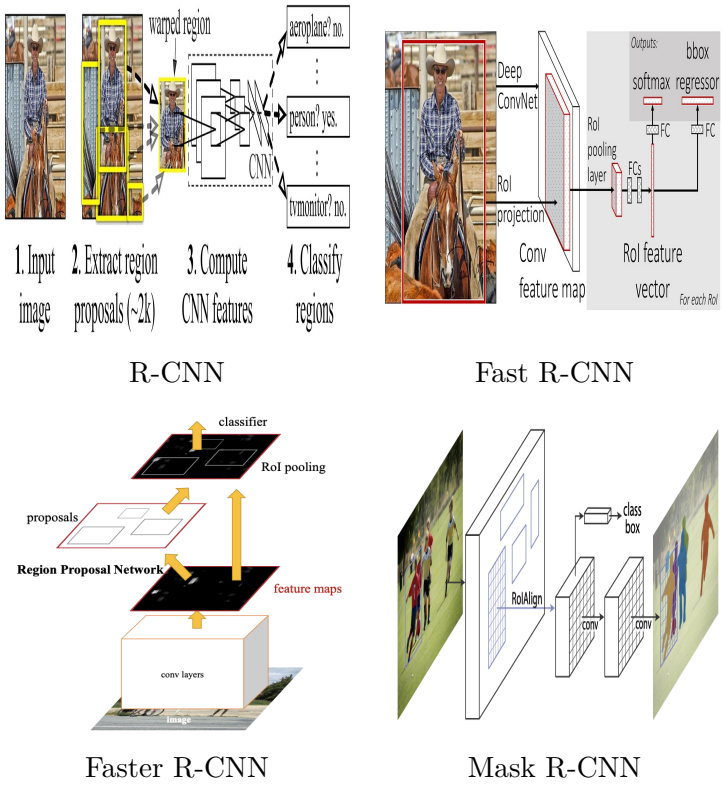


Figure 1.22: the family of R-CNN model architectures

Additionally, when trained on the COCO trainval35k dataset, the SSD300 and SSD512 models achieve mAP scores of 41.2% and 46.5%, respectively.

The SSD512 model, trained on the COCO trainval35k dataset and fine-tuned on Pascal VOC 2007+2012, achieves the highest performance with a mean average precision (mAP) of 81.6%. In contrast, the SSD300 model, trained solely on the VOC 2007 dataset, achieves a lower mAP of 68.0%. Additionally, when trained on the COCO trainval35k dataset, the he achieve mAP scores of 41.2% and SSD512 46.5% of mAP.

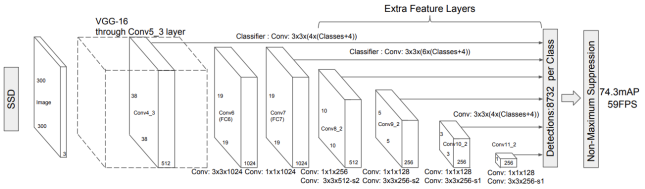


Figure 1.23: the model's SSD architecture [4]

1.7.6.6 Detr

Detr, or Detection Transformer, developed by Nicolas Carion et al. [5] is a set-based object detector using a transformer-based model on top of a convolutional backbone originally designed for natural language processing to address object detection. By treating object detection as a set prediction problem and using learned object queries, DETR uses the conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder, to accurately identify

and localize objects within an image, simplifying the object detection pipeline and achieving competitive performance without the need for anchor boxes or region proposal networks. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call object queries, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a “no object” class.

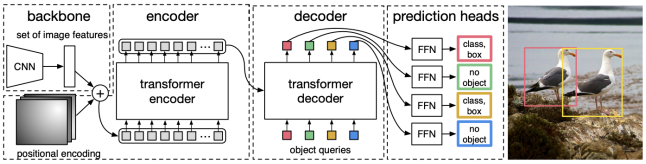


Figure 1.24: Detr model architecture [5]

1.7.6.7 RetinaNet

RetinaNet is a one-stage object detection model introduced in 2017 by Facebook AI Research (FAIR), that utilizes a focal loss function to address class imbalance during training. RetinaNet extends Faster R-CNN by introducing an additional branch known as the "classification subnetwork." This innovative approach, developed by Tsung-Yi Lin et al. in 2017 [6], tackles class imbalance and handles objects of varying scales and aspect ratios.

It uses a backbone CNN (e.g., ResNet) for feature map extraction and a Feature Pyramid Network for multi-scale feature maps. With two subnetworks for classification and regression, it predicts object presence, class, and bounding boxes. The first subnet performs convolutional object classification on the backbone’s output; the second subnet performs convolutional bounding box regression. The two subnetworks feature a simple design that the authors propose specifically for one-stage, dense detection.

The classification subnetwork, a fully convolutional network (FCN) applied to a fixed-size feature map, generates multiple anchor boxes per location. It provides class probabilities and bounding box offsets for each anchor, enabling precise object localization. Unlike Fast/Faster R-CNN, which predicts a single bounding box and class label, RetinaNet predicts object class probabilities for each anchor, resulting in improved accuracy. RetinaNet achieves remarkable validation results, when evaluated on the MS COCO + fine database using ResNet-50-FPN, including an Average Precision (AP) of 39.1%, AP50 of 59.1%, AP75 of 42.3%. The exceptional performance of RetinaNet in addressing challenges such as class imbalance and object scale variance firmly establishes it as a state-of-the-art object detection model [6].

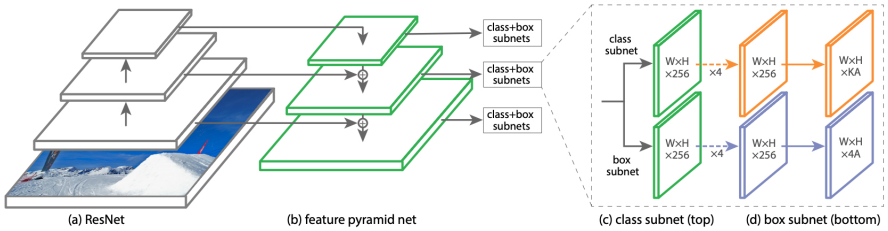


Figure 1.25: RetinaNet model architecture [6]

1.7.6.8 YOLO Model

The YOLO (You Only Look Once) algorithm, initially introduced in the paper "You Only Look Once: Unified, Real-Time Object Detection" by Joseph Redmon et al. in June 2016 [7], has transformed object detection with its efficient and precise methodology. Unlike traditional approaches, YOLO directly predicts bounding boxes and class probabilities in a single evaluation, removing the need for region proposals. By analyzing features from the entire image, the network comprehensively assesses all objects, resulting in accurate predictions. Simultaneously predicting bounding boxes for all classes enhances detection efficiency and coherence. YOLO's end-to-end learning enables joint training of the entire detection pipeline (refer to Figure 1.26), leading to faster inference times. Impressively, despite its real-time speed, YOLO maintains high average precision, making it well-suited for various applications.

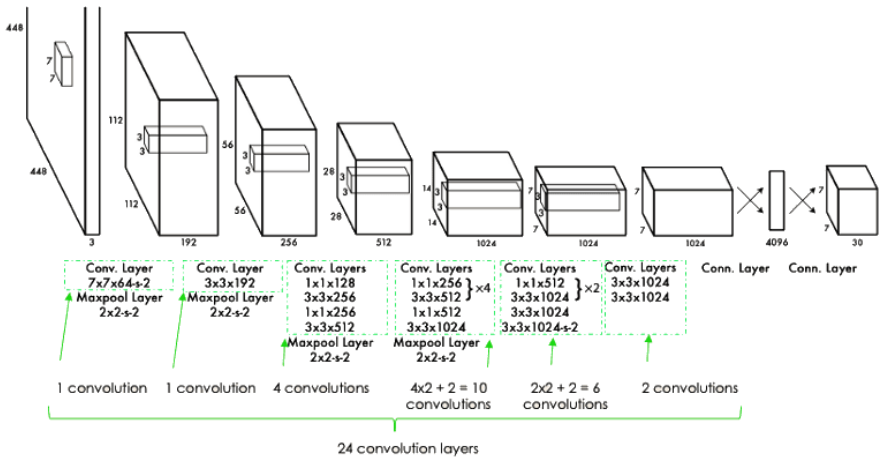


Figure 1.26: YOLO model architecture [7]

The network architecture comprises 24 convolutional layers and 2 fully connected layers. It draws inspiration from the GoogLeNet model for image classification but differs in terms of initialization modules. Instead, YOLO incorporates 1x1 reduction layers followed by 3x3 convolutional layers. This alteration allows YOLO to achieve real-time object detection without compromising accuracy, presenting a unified approach to the task.

The YOLO algorithm model is divided into three stages:

- **Image Division into Cells:**
  The image is divided into a grid of size $S \times S$, such as 3 x 3 in this example, resulting

in a total of N cells. Each cell in the grid is tasked with detecting objects within its designated region.

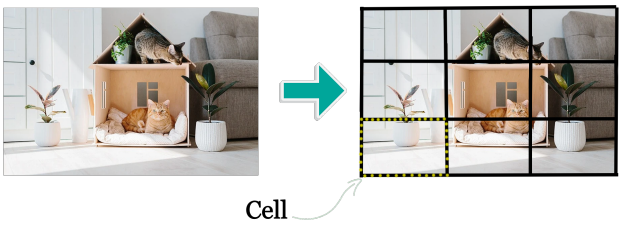


Figure 1.27: Divide the image into (S*S) grid.

- **Each cell predicts B bounding boxes:**
  After dividing the image into N cells, each grid cell predicts B bounding boxes and associated confidence scores. These bounding boxes comprise five predictions: x, y, w, h, and confidence. The coordinates (x, y) denote the center of the box relative to the grid cell's boundaries, while the width and height are predicted relative to the entire image. The confidence prediction signifies the Intersection over Union (IoU) between the predicted box and any ground truth box [7].

  For instance, in a scenario where the image is segmented into a 3x3 grid (S=3), each cell predicts a single bounding box (B=1). The objects in the image, whether horses (represented as 1) or humans (represented as 2), are identified by the CNN, which produces a vector Y for each cell (see Figure 1.28).

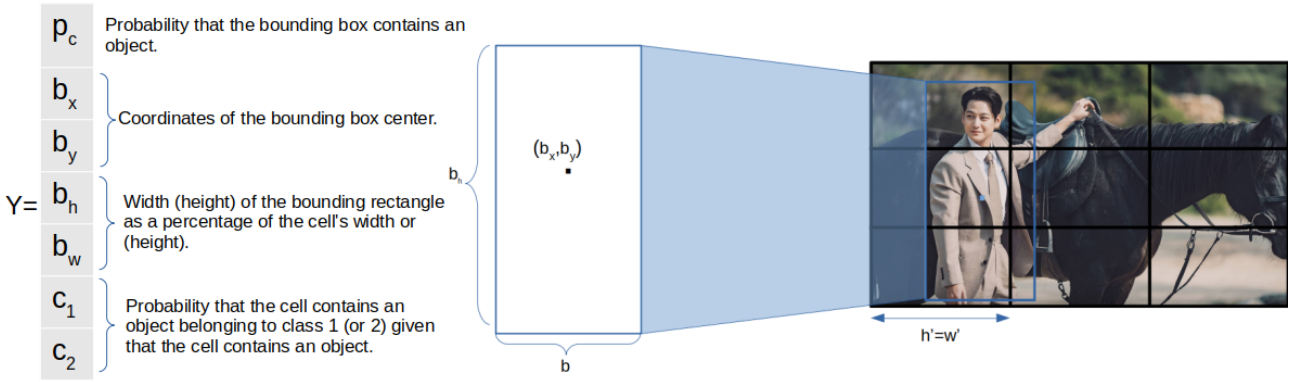


Figure 1.28: The predicted vector in the case of a single box.

The values within vector Y adhere to the YOLO format, wherein P_c denotes the confidence prediction representing the Intersection over Union (IoU) between the predicted box and the ground truth box. The expressions for bx, by, bh, and bw are derived as follows:

$$
\begin{aligned}
bx &= \frac{(x - h')}{h'}, \\
by &= \frac{(y - w')}{w'}, \\
bh &= \frac{h}{416}, \\
bw &= \frac{w}{416}.
\end{aligned}
$$

– **Intersection over Union (IoU):** Intersection over Union (IoU), also referred to as Intersection over Union, stands as a prevalent evaluation metric in object detection. It functions as a gauge to assess the accuracy of predictions by contrasting the overlap between the predicted bounding box and the ground truth box. To gauge performance, a precision threshold is designated [45]. The IoU, illustrated in Figure 1.29, calculates the ratio of the area of intersection to the area of union, furnishing crucial insights into the quality of object detection predictions.

Figure 1.29: IoU : Intersection over union.

Throughout the training process, the confidence of the predicted bounding box undergoes evaluation by computing the Intersection over Union (IoU) score with the ground truth box. In Figure 1.30, showcased examples depict both high and low IoU scores. Clearly, predicted bounding boxes with significant overlap with the ground truth boxes attain higher scores, while those with lesser overlap garner lower scores.

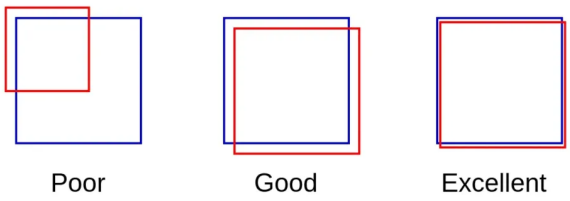


Figure 1.30: Examples of IoU.

– **Anchor Box :** In the preceding example, we discussed the scenario where only one bounding box was predicted. However, in situations where multiple bounding boxes

exist within the same grid cell, YOLO addresses this by utilizing anchor boxes. As previously mentioned, each cell is represented by a vector. When multiple boxes are present in a cell, the vector is adjusted accordingly to accommodate the additional bounding boxes.

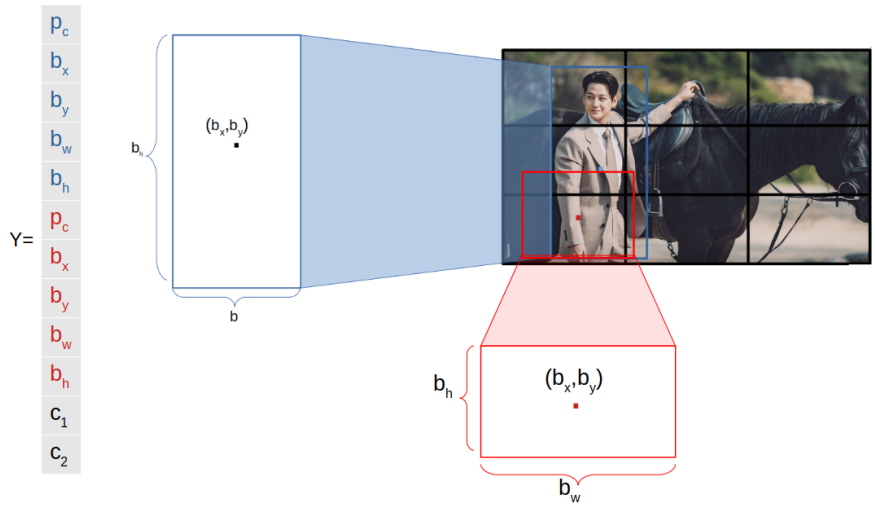


Figure 1.31: predicted vector when multiple boxes.

In a general sense, when we partition the image into a grid of size $S \times S$, each individual cell within the grid predicts B bounding boxes, their corresponding confidence scores, and the class probabilities C. These predictions are then encoded as a tensor with dimensions [7].

– **Non-Maximum Suppression :** Non Maximum Suppression is a computer vision method (introduced by [40]) that selects a single entity out of many overlapping entities (for example bounding boxes in object detection). The criteria is usually discarding entities that are below a given probability bound. With remaining entities we repeatedly pick the entity with the highest probability, output that as the prediction, and discard any remaining box where a with the box output in the previous step. Non-Maximum Suppression (NMS) serves as the final step in the detection algorithm, activated when multiple bounding boxes detect the same object in the image. Its aim is to eliminate less probable bounding boxes and preserve only the most accurate one. This technique involves a series of five steps to accomplish this goal [8].

  * **Step 1:** Choose the box with the highest confidence score as the initial selection.
  * **Step 2:** Evaluate the overlap (Intersection over Union) between this selected box and other boxes.
  * **Step 3:** Eliminate bounding boxes with an overlap greater than 50%.
  * **Step 4:** Move on to the next box with the highest confidence score.
  * **Step 5:** Repeat steps 2 to 4 until all objects in the image have been processed.

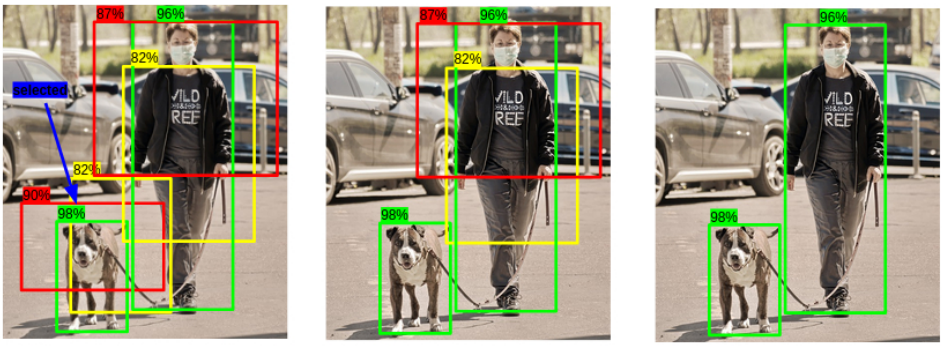


Figure 1.32: Output after NMS steps [8]

1.7.6.9 YOLOv2 Model

In December 2016, Joseph Redmon and Ali Farhadi enhanced the YOLO model for efficiency, speed, and robustness [46]. They utilized Darknet-19 with 30 layers, added batch normalization, and raised the input resolution to 448. The model employed a grid system with 5 bounding boxes per grid cell and anchor boxes for precise localization, resulting in a more powerful YOLO version then YOLOv1.

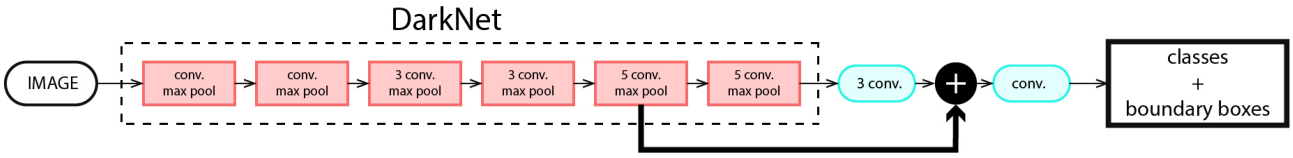


Figure 1.33: Used Darknet in YOLOv2

1.7.6.10 YOLOv3 Model

In 2018, the same authors of yolov2 improved upon the previous version with several enhancements. They introduced **YOLOV3** with the new backbone Darknet53 [9] for feature extraction and employed logistic regression to compute an objectness score for each bounding box. To address class predictions, they utilized binary cross-entropy loss. Unlike YOLOv2, which struggled with detecting small objects, this version tackled the issue by representing boxes at three different scales.

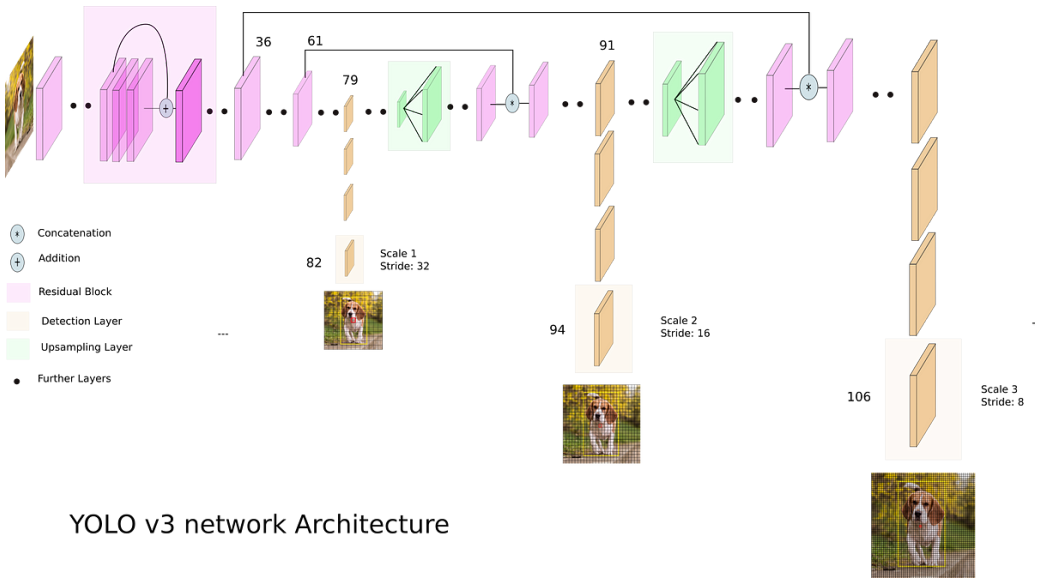


Figure 1.34: YOLOv3 Architecture [9]

1.7.6.11 YOLOv4 Model

YOLOv4, introduced in the paper titled "YOLOv4: Optimal Speed and Accuracy of Object Detection" by Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao in 2020 [47], aims for optimal speed and accuracy in object detection. It enhances previous models with advanced backbone networks like CSPDarknet53 and introduces components like SPP (Spatial Pyramid Pooling) and PANet (Path Aggregation Network) for improved feature extraction and fusion. To optimize speed and accuracy, it incorporates techniques such as data augmentation, optimization algorithms, and varied anchor boxes for detecting objects of different sizes.

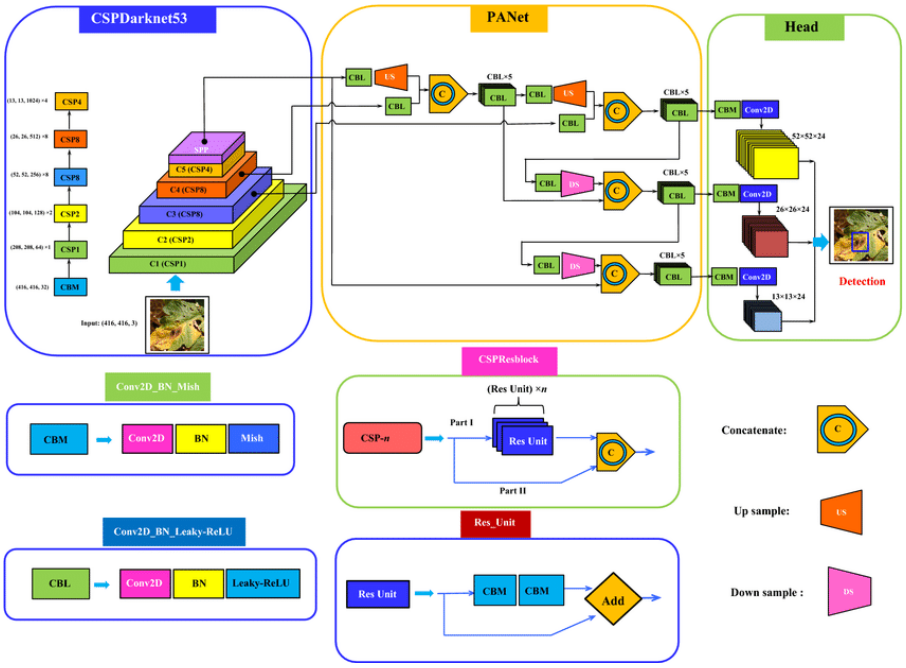


Figure 1.35: YOLOv4 Architecture

1.7.6.12 YOLOv5 Model

YOLOv5, developed by Glen Jocher in 2021 [48], founder and CEO of Ultralytics, builds upon the advancements of YOLOv4 and is implemented using PyTorch. Its architecture features a modified CSPDarknet53 backbone, a neck utilizing SPPF and a modified CSP-PAN, and a head similar to YOLOv3. Key enhancements include the AutoAnchor algorithm for adjusting anchor boxes, along with augmentations like Mosaic, random affine, MixUp, HSV augmentation, and improved grid sensitivity. YOLOv5 offers multiple scaled versions (YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x) catering to various application needs and hardware requirements, with lightweight models for low-resource devices and high-performance models optimized for speed. Overall, YOLOv5 delivers improved object detection capabilities, real-time performance, and a balance between accuracy and speed, supported by Ultralytics with active maintenance and a thriving community of contributors.

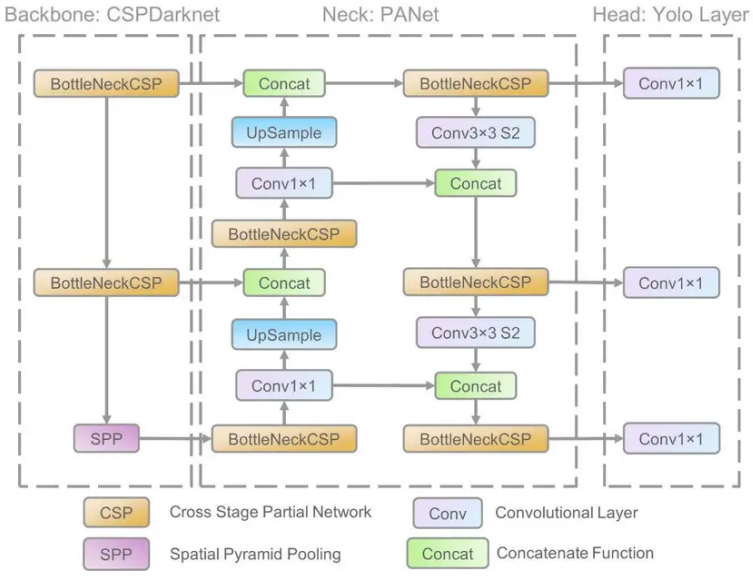


Figure 1.36: YOLOv5 Architecture

1.7.6.13 YOLOv6 Model

YOLOv6, developed by the Meituan Vision AI Department in September 2022 [10]. Its architecture consists of three main elements: the EfficientRep backbone, built on RepVGG for enhanced parallelism; a PAN topology neck incorporating PAN with RepBlocks or CSPStackRep Blocks for larger models; and an efficient decoupled head inspired by YOLOX. YOLOv6 introduces novel features like Task alignment learning for label assignment, upgraded classification and regression losses (VariFocal loss and SIoU/GIoU loss), and self-distillation techniques for improved regression and classification tasks. Additionally, it implements a quantization scheme using RepOptimizer and channel-wise distillation, significantly boosting detection speed without compromising accuracy. YOLOv6-L achieves remarkable results, with an AP of 52.5% and AP50 of 70% on the MS COCO dataset's test-dev 2017, while maintaining an operational speed of around 50 Frame Per Second on NVIDIA Tesla T4 GPU.

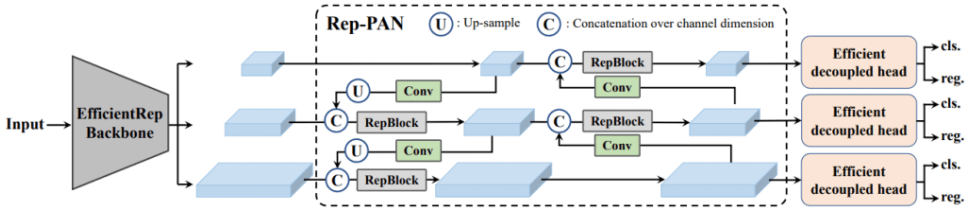


Figure 1.37: YOLOv6 Architecture [10]

1.7.6.14 YOLOv7 Model

YOLOv7 [11], introduced in July 2022 by the developers of YOLOv4 and YOLOR, distinguished itself with exceptional speed and accuracy across diverse object detection scenarios, achieving speeds ranging from 5 FPS to 160 FPS. In contrast to its predecessors, YOLOv7 was trained exclusively on the MS COCO dataset without the reliance on pre-trained backbones. It brought forth significant architectural modifications and a suite of "bag-of-freebies" techniques aimed at boosting accuracy while preserving inference speed, albeit with an increase in training duration. These changes included adopting the Extended Efficient Layer Aggregation Network (E-ELAN) for more efficient model convergence and implementing model scaling for concatenation-based architectures to maintain optimal structure while adjusting size. Additionally, YOLOv7 integrated various enhancements like planned re-parameterized convolution (RepConvN), coarse and fine label assignment strategies, batch normalization during inference, implicit knowledge from YOLOR, and employing exponential moving average for the final inference model. Together, these enhancements significantly improved YOLOv7's performance in terms of accuracy and speed.

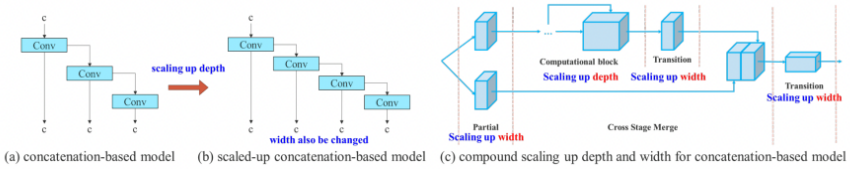


Figure 1.38: YOLOv7 Architecture [11]

1.7.6.15 YOLOv8 Model

Introduced by Ultralytics in January 2023 [49], **YOLOv8** shares similarities with YOLOv5 in its architecture, featuring backbone, head, and neck components. However, YOLOv8 introduces several enhancements, including improved convolutional layers in the backbone and an advanced detection head, positioning it as a top-tier solution for real-time object detection tasks. It also supports various computer vision algorithms, such as instance segmentation, enabling the detection of multiple objects in images or videos.

YOLOv8 utilizes the Darknet-53 backbone network, which offers improved speed and accuracy compared to its predecessor, YOLOv7. Employing an anchor-free detection head for predicting bounding boxes, YOLOv8 achieves enhanced effectiveness. Furthermore, the model

benefits from a larger feature map and an optimized convolutional network, enhancing both precision and speed. Additionally, YOLOv8 incorporates feature pyramid networks to effectively detect objects of varying sizes.

Moreover, YOLOv8 provides a user-friendly API, simplifying its integration across diverse applications. With its blend of architectural improvements, upgraded convolutional layers, and advanced detection capabilities, YOLOv8 emerges as a powerful solution for real-time object detection tasks [12].

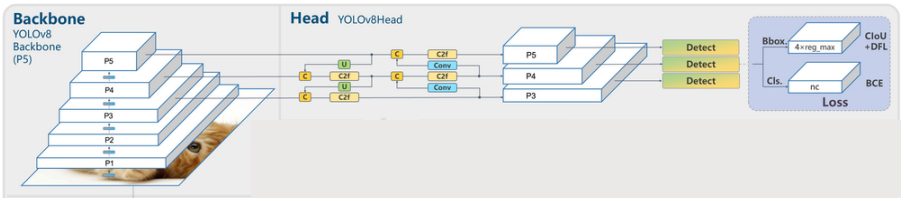


Figure 1.39: YOLOv8 Architecture [12]

1.7.6.16 YOLO NAS Model

YOLO-NAS stands as a cutting-edge object detection model developed by Deci AI [50], building on prior research [13]. It surpasses YOLOv6 and YOLOv8 in mean average precision (mAP) and inference latency, pre-trained on datasets like COCO and Objects365 for practical applications. Accessible via Deci's SuperGradients library, YOLO-NAS leverages Neural Architecture Search (NAS) with Deci's AutoNAC technology to determine optimal network architectures automatically. This involves configuring stage sizes, block types and quantities, and channel numbers, considering hardware and data-awareness for accuracy, computational complexity, and model size optimization.

The YOLO-NAS architecture integrates Quantization-Aware RepVGG (QA-RepVGG) blocks, ensuring compatibility with Post-Training Quantization (PTQ) for 8-bit quantization and reparameterization benefits, minimizing accuracy loss. Employing hybrid quantization selectively optimizes accuracy and latency tradeoffs while maintaining overall performance. Furthermore, attention mechanisms and inference time reparametrization further enhance YOLO-NAS's real-time object detection capabilities, contributing to its exceptional performance.

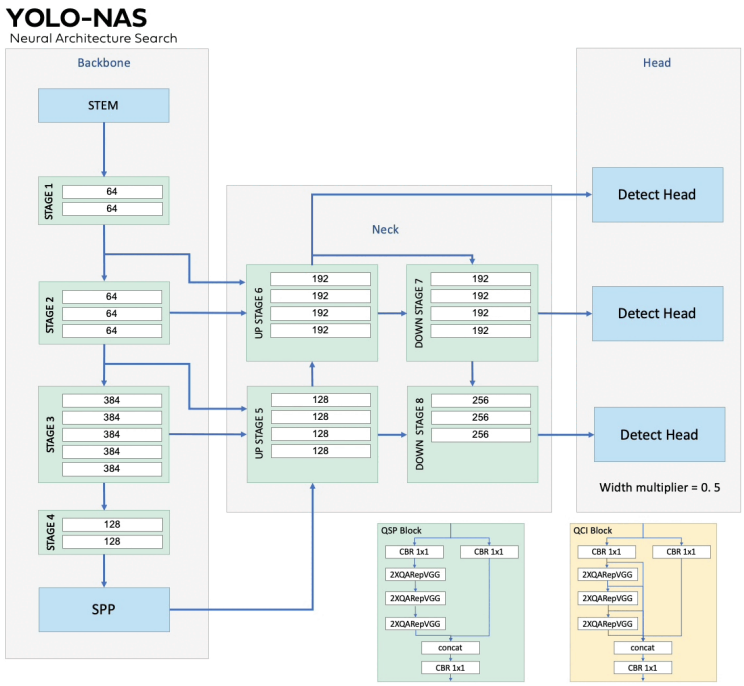


Figure 1.40: YOLO NAS Architecture [13]

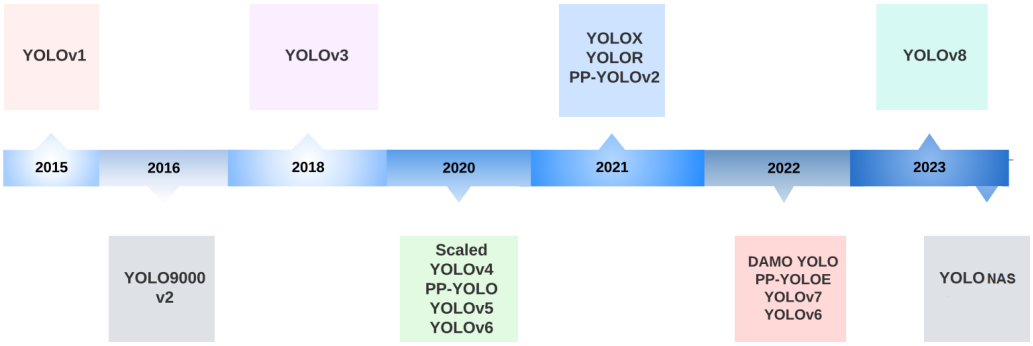


Figure 1.41: YOLO timeline Architectures

1.7.6.17 What makes YOLO a better choice for Object Detection ?

renowned for its real-time performance and accuracy, is a widely-used object detection model. It's worth noting the various versions of YOLO, including YOLOv1 through YOLONAS, each bringing enhancements and modifications to the original architecture. These advancements significantly bolster YOLO's accuracy and performance in object detection tasks. The main distinctions and advantages of YOLO compared to other object detection models include:

- Single-shot detection: YOLO conducts object detection and classification in one pass, improving efficiency.
- Speed: YOLO is designed for real-time detection, dividing images into a grid for rapid processing.
- Simplicity: YOLO's architecture is straightforward, simplifying training and deployment.

- Multi-scale predictions: YOLO detects objects of various sizes by using feature maps from different network layers.
- High accuracy: YOLO achieves competitive accuracy on benchmark datasets, balancing speed and precision.
- Objectness score: YOLO's "objectness" score filters out low-confidence detections, reducing false positives.
- Trade-off between localization and classification: YOLO optimizes a joint loss function for comprehensive object detection.

1.8 Datasets(Benchmarks)

In the realm of machine learning algorithms, the acquisition of data holds immense importance as it facilitates AI's understanding of human cognition. Furthermore, data expedites the learning curve, leading to enhanced accuracy as the algorithm acquires more knowledge. The widespread availability of databases has transformed the landscape of machine learning models, simplifying their development and bolstering their effectiveness.

This section will delve into benchmark databases utilized for detection tasks, outlining their distinctive attributes and the prevalent models employed within each database.

1.8.1 MS COCO dataset :

MS COCO, short for Microsoft Common Objects in Context, stands as a specialized dataset tailored for object detection, segmentation, and captioning within computer vision. Comprising a vast collection, it boasts over 330,000 images meticulously annotated with 2.5 million object instances spanning 80 distinct categories. The annotations encompass detailed object bounding boxes, segmentation masks, and descriptive captions. Renowned for its comprehensive coverage and meticulously crafted annotations, MS COCO serves as a prominent benchmark dataset extensively utilized for training and assessing deep learning models across a spectrum of computer vision tasks [51].

Figure 1.42: Example of images of MS-COCO

Example of annotation of MS-COCO:

```
{
    "image_id": 12345,
    "category_id": 17,
    "bbox": [100, 50, 200, 150],
    "segmentation": [[100, 50, 300, 50, 300, 200, 100, 200]],
    "area": 30000,
    "iscrowd": 0
}
```

In this example:

"image_id" represents the unique identifier of the image. *"category_id"* denotes the category label of the annotated object. *"bbox"* specifies the bounding box coordinates in the format [x, y, width, height]. *"segmentation"* outlines the segmentation mask of the annotated object. *"area"* indicates the area of the annotated object. *"iscrowd"* is a flag (0 or 1) indicating whether the annotated object is a single object or a group of objects.

This annotation describes an object instance with the category ID 17 (e.g., "person") located at the bounding box [100, 50, 200, 150] and corresponding segmentation mask defined by the polygon coordinates [[100, 50], [300, 50], [300, 200], [100, 200]].

Table 1.5 shows the top 5 Object Detection Models on COCO Dataset with their Mean Average Precision (mAP).

Model Name	Mean Average Precision (mAP)
YOLOv4	48.5%
EfficientDet	47.2%
Faster R-CNN	45.8%
YOLOv3	43.0%
Mask R-CNN	42.3%

Tableau 1.5: Top 5 Object Detection Models on COCO Dataset

1.8.2 PASCAL VOC dataset :

PASCAL VOC (Visual Object Classes) stands as a prominent computer vision dataset widely embraced for tasks like object detection, segmentation, and classification. This dataset encompasses annotations of objects within images, comprising a comprehensive range of 20 object classes spanning animals, vehicles, household items, and more. Annotations within PASCAL VOC entail bounding boxes surrounding each object instance, accompanied by corresponding object class labels.

Divided into training and validation segments, the dataset comprises approximately 11,000 images in the training set and 5,000 images in the validation set. Both sets are meticulously annotated with object instance segmentation masks, bounding boxes, and class labels.

Renowned as a benchmark in the field, PASCAL VOC has significantly influenced the development of object detection and segmentation algorithms. Its widespread utilization in

research has led to the evaluation of numerous state-of-the-art models, cementing its pivotal role in advancing computer vision technologies [52].

```
<annotation>
        <folder>Kangaroo</folder>
        <filename>00001.jpg</filename>
        <path>./Kangaroo/stock-12.jpg</path>
        <source>
                <database>Kangaroo</database>
        </source>
        <size>
                <width>450</width>
                <height>319</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>kangaroo</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>233</xmin>
                        <ymin>89</ymin>
                        <xmax>386</xmax>
                        <ymax>262</ymax>
                </bndbox>
        </object>
</annotation>
```

Figure 1.43: Example of annotation of PASCAL VOC

the annotation 1.43 for an image from the Pascal VOC dataset includes information about the folder, filename, source, image size, segmentation, and details about the object detected in the image (in this case, a person).

Table 1.6 shows the top 5 Object Detection Models on Pascal VOC Dataset with their Mean Average Precision (mAP).

Model Name	Mean Average Precision (mAP)
SSD	77.1%
RetinaNet	76.5%
Faster R-CNN	74.3%
YOLOv3	71.8%
YOLOv4	70.7%

Tableau 1.6: Top 5 Object Detection Models on Pascal VOC Dataset

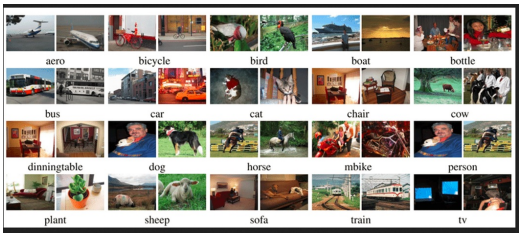


Figure 1.44: Images from PASCAL VOC

1.8.3 ImageNet dataset

The ImageNet dataset is renowned for its vast collection of meticulously annotated images, comprising over 14 million labeled instances spanning a staggering 21,841 distinct object categories or classes. This expansive dataset is meticulously organized hierarchically, featuring subclasses and superclasses that offer a nuanced and comprehensive representation of various objects and their intricate relationships. The meticulous labeling of images within this dataset facilitates precise model training and evaluation, making it an invaluable resource for advancing the field of deep learning in computer vision. Furthermore, ImageNet's extensive coverage across a diverse array of object categories enables the development and evaluation of deep learning models capable of recognizing and classifying a wide spectrum of objects and concepts depicted in images. This rich diversity and meticulous organization contribute to ImageNet's pivotal role as a benchmark dataset for assessing the performance and robustness of image classification models across various domains and applications in the realm of computer vision.

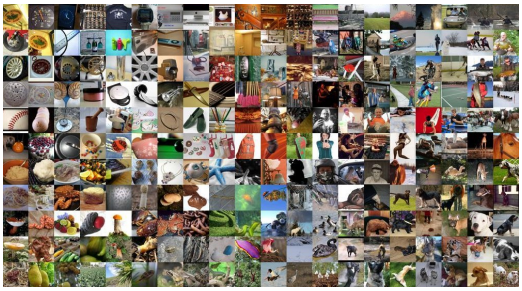


Figure 1.45: Images from ImageNet

Table 1.7 shows the top 5 Object Detection Models on ImageNet Dataset with their Mean Average Precision (mAP).

Model Name	Mean Average Precision (mAP)
Faster R-CNN	70.3%
R-FCN	69.8%
YOLOv4	68.5%
SSD	67.2%
YOLOv3	66.4%

Tableau 1.7: Top 5 Object Detection Models on ImageNet Detection Dataset

1.8.4 Open Images dataset

Open Images [53] is a vast dataset consisting of 9,178,275 images and extensive annotations, including 30,113,078 image-level labels, 15,440,132 bounding boxes, and 374,768 visual relationship triplets. It also encompasses 19,794 image-level labels and 600 bounding box categories, making it a valuable resource for advancing data-hungry methods in the field. Notably, the dataset offers unparalleled scale in object detection annotations, with 15.4 million bounding

boxes covering 600 categories across 1.9 million images. This dataset is accessible through approximately 9 million annotated image URLs stored in a CSV file and includes over 6,000 categories. It is subdivided into training, validation, and testing subsets to facilitate comprehensive evaluation. For the most precise and up-to-date information about the Open Images dataset, it is advisable to consult the official Open Images website or relevant research papers.

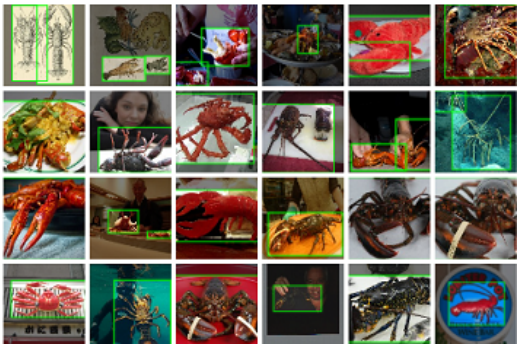


Figure 1.46: Images from Open Image

Table 1.8 shows the top 5 Object Detection Models on Open Image Dataset with their Mean Average Precision (mAP).

Model Name	Mean Average Precision (mAP)
EfficientDet	56.3%
YOLOv5	54.9%
RetinaNet	53.5%
SSD	51.8%
YOLOv4	49.7%

Tableau 1.8: Top 5 Object Detection Models on Open Images Dataset

1.8.5 summary of Benchmark Datasets for Object Detection

table 1.9 shows a summary of some of the most used datasets in the field of object detection.

Dataset Name	Description	Number of Images	Number of Classes	Number of Instances	Image Size	Year
MS COCO	Common Objects in Context	207K	80	1.5M	Various	2014
Pascal VOC	Visual Object Classes	11.5K	20	27K	512x512	2005
ImageNet Detection	ImageNet Large Scale Visual Recognition Challenge	1.2M	200	N/A	Various	N/A
Cityscapes	Semantic understanding of urban street scenes	5K (Annotated)	30	N/A	2048x1024	2016
Open Images	Collaboratively Annotated Images	9M	600+	N/A	Various	2016
ADE20K	ADE20K Scene Parsing	25K	150	N/A	Various	2017
KITTI	KITTI Vision Benchmark Suite	7K (Annotated)	3	N/A	1242x375	2012
Dota	DOTA: A Large-scale Dataset for Object Detection in Aerial Images	2.8K	16	N/A	Various	2018
Caltech	Caltech Pedestrian Detection	10K	2	N/A	640x480	2009
CIFAR	CIFAR-10/CIFAR-100	60K (combined)	10/100	N/A	32x32	N/A

Tableau 1.9: Benchmark Datasets for Object Detection

1.8.6 Challenges in Data Preparation

When collecting and processing data for machine learning tasks, several crucial characteristics must be considered. Here are some key points to keep in mind:

- **Images per Class:** It is essential to ensure that the model receives an adequate number of training examples for each class, enabling it to learn and accurately recognize objects across all categories. It is recommended to have at least 1500 images per class, and if this threshold is not met, data augmentation techniques, such as those used for indoor datasets, can be employed.
- **Instances per Class:** Providing the model with a sufficient number of examples for each class during training is crucial. The recommended number of instances (labeled objects) per class varies depending on the complexity of the object.
- **Image Variety:** To create a dataset that accurately represents the environment where the model will be deployed, it is important to include images that capture various factors such as different times of day, seasons, weather conditions, lighting conditions, angles, and sources (e.g., obtained online, collected locally, or from different cameras).
- **Label Consistency:** Complete labeling is necessary for all instances of all classes in every image. Partial labeling is insufficient for the task.
- **Label Accuracy:** Labels should tightly encompass each object, leaving no gaps between the object and its bounding box. Each object should have a corresponding label, and no objects should lack a label.
- **Label Verification:** To ensure label accuracy, it is recommended to review the `train_batch*.jpg` images at the beginning of the training process. This allows for verification of correct label application and identification of anomalies, such as examining example mosaics.
- **Background Images:** Background images, which do not contain any objects, are included in a dataset to minimize the occurrence of False Positives (FP). It is advisable to include approximately 0-10% of background images to effectively reduce FPs. For reference, the COCO dataset includes 1000 background images, constituting 1% of the total dataset. Background images do not require any labels.
- **Intercalss Variety:** One of the primary challenges in object detection is dealing with datasets with low interclass variance, where multiple classes resemble each other more closely compared to other labels. For instance, classes like 'laptop' and 'tvmonitor' present such a challenge.

1.8.7 a leaderboard for object detection models on different datasets

here is in table 1.10 leaderboard with Mean Average Precision (mAP) values for object detection models on different datasets::

Dataset	Model Name	Mean Average Precision (mAP)	FPS	Image Size (Resolution)	Number of Images	Number of Classes	Number of Instances	Year
COCO	YOLOv4	48.5%	30 FPS	Various	207K	80	1.5M	2014
COCO	EfficientDet	47.2%	40 FPS	Various	207K	80	1.5M	2014
COCO	Faster R-CNN	45.8%	25 FPS	Various	207K	80	1.5M	2014
Pascal VOC	SSD	77.1%	90 FPS	512x512	11.5K	20	27K	2005
Pascal VOC	RetinaNet	76.5%	80 FPS	512x512	11.5K	20	27K	2005
ImageNet Detection	Faster R-CNN	70.3%	35 FPS	Various	1.2M	200	N/A	N/A
ImageNet Detection	R-FCN	69.8%	30 FPS	Various	1.2M	200	N/A	N/A
Cityscapes	DeeplabV3+	75.2%	N/A	2048x1024	5K (Annotated)	30	N/A	2016
Cityscapes	ENet	71.8%	N/A	2048x1024	5K (Annotated)	30	N/A	2016
Open Images	EfficientDet	56.3%	45 FPS	Various	9M	600+	N/A	2016
Open Images	YOLOv5	54.9%	35 FPS	Various	9M	600+	N/A	2016

Tableau 1.10: Leaderboard of Object Detection Models on Different Datasets

1.9 Frameworks for Deep Learning

There exists a broad array of popular frameworks, offering user-friendly interfaces for constructing, training, and deploying deep neural networks. Below are some widely recognized frameworks in this domain:

(a) **TensorFlow (2015):** TensorFlow stands out as one of the most extensively used and accepted frameworks for deep learning applications. It boasts a comprehensive ecosystem featuring a flexible architecture and multi-language support including Python, C++, and JavaScript. TensorFlow offers both high-level APIs like Keras, and low-level APIs for advanced customization.

(b) **PyTorch (2016)** ***"caffe2 was merged into PyTorch in 2018"*****:** PyTorch is another prevalent framework known for its dynamic computational graph, enhancing flexibility and user intuitiveness. Renowned for its simplicity and swift experimentation capabilities, PyTorch is predominantly utilized with Python.

(c) **Keras (2015):** Originally an independent project, Keras is now integrated into the TensorFlow ecosystem. It presents a user-friendly, high-level API for constructing neural networks, prioritizing simplicity and a modular, extendable interface. Keras supports both TensorFlow and Theano as backend engines.

(d) **Caffe (2013):** Caffe, which stands for Convolutional Architecture for Fast Feature Embedding, prioritizes speed and efficiency, particularly in computer vision tasks. Noted for its expressive architecture definition, Caffe is implemented in C++ with a Python interface available.

(e) **MXNet (2015):** MXNet, an open-source framework, is recognized for its scalability and efficiency. Offering a flexible programming interface, it supports multiple languages such as Python, R, Scala, and Julia. MXNet provides both symbolic and imperative APIs, designed for distributed computations across various devices and machines.

(f) **Theano (2007):** Theano, a popular choice in deep learning, excels in efficiently computing mathematical expressions involving multi-dimensional arrays. It was among the pioneering frameworks enabling researchers to define, optimize, and evaluate mathematical expressions effectively, primarily used with Python.

(g) **Microsoft Cognitive Toolkit (CNTK) (2016):** Developed by Microsoft, CNTK offers a flexible architecture emphasizing scalability and performance for building diverse deep learning models. Supporting both high-level and low-level APIs, CNTK is compatible with Python, C++, and C#.

(h) **PyTorch Lightning (2019):** PyTorch Lightning is a lightweight, modular wrapper around PyTorch, simplifying the training and research workflow. It offers a high-level interface featuring pre-built training loops, automatic distributed training handling, and utilities for improved organization and readability of deep learning code.

Additionally, numerous other frameworks such as Matlab Deep Learning Toolbox, Apple CoreML (2017), ML.NET (2002), Dlib (2002), Torch (2002), PlaidML (2017), Deeplearning4j (2014), Chainer (2015), BigDL (2016), Flux framework (2017), and Open Neural Network Exchange (ONNX) (2017) exist.

These frameworks offer diverse features, and the selection of the most suitable one depends on factors like task complexity, personal preferences, community support, and deployment requirements. Each framework possesses its unique strengths and weaknesses, making it beneficial to explore and experiment with different options to find the best fit for specific needs.

there are so many other frameworks: Matlab deeplearning toolbox, Apple CoreML (2017), ML.NET (2002), Dlib(2002), Torch (2002), PlaidML(2017), Deeplearning4j (2014), Chainer (2015), BigDL (2016), Flux framework (2017), Open Neural Network Exchange (ONNX) (2017).

1.10 Taxonomy

1.10.1 Taxonomy of object detection methods

We provide tables of possible taxonomy of object detection methods using artificial intelligence, categorized based on various factors, to help this taxonomy:

Factor 1: Network Architecture

Category	Explanation	Examples
Single Shot	Single-shot detectors are one-stage object detection methods that predict bounding boxes and class labels in a single pass through the network. They are known for their real-time processing capabilities and are suitable for applications where speed is critical.	YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector)
Two-Stage	Two-stage detectors first propose regions of interest in the image and then classify and refine these regions. They are typically more accurate but can be computationally intensive.	Faster R-CNN, R-FCN (Region-based Fully Convolutional Networks)
Region-based	These methods use region proposals to identify objects. They typically involve a region proposal network (RPN) to generate potential object regions and then use these regions for classification and localization.	Fast R-CNN, Mask R-CNN
Anchor-based	Anchor-based detectors use predefined anchor boxes of various sizes and aspect ratios to predict objects' locations and classes within these anchor boxes.	RetinaNet, CenterNet
Anchor-free	Anchor-free detectors do not rely on predefined anchor boxes. Instead, they directly predict object locations and class labels without using anchors, which can lead to more flexibility in object localization.	CornerNet, FCOS (Fully Convolutional One-Stage Object Detection)

Factor 2: Backbone Network

Category	Explanation	Examples
VGG-based	VGGNet is a popular convolutional neural network (CNN) architecture. VGG-based backbones are used in some object detection models for feature extraction.	VGG16, VGG19, VGG16-BN
ResNet-based	ResNet (Residual Network) architectures are known for their deep structures and skip connections. Many modern object detection models use ResNet-based backbones for their powerful feature extraction capabilities.	ResNet-50, ResNet-101, ResNet-152, ResNet-18, ResNet-34
MobileNet-based	MobileNet is designed for mobile and embedded vision applications. MobileNet-based backbones are used to reduce computational complexity while maintaining accuracy.	MobileNetV2, MobileNetV3, MobileNetV1
EfficientNet-based	EfficientNet is an architecture that balances model size and accuracy. Object detection models use EfficientNet-based backbones to achieve efficient and effective feature extraction.	EfficientNetB0, EfficientNetB1, EfficientNetB2, EfficientNetB3, EfficientNetB4, EfficientNetB5, EfficientNetB6, EfficientNetB7
DenseNet-based	DenseNet (Densely Connected Convolutional Networks) is a neural network architecture where each layer is connected to every other layer in a feed-forward fashion. Dense connections enable the network to reuse features from multiple layers, which helps in feature propagation and gradient flow.	DenseNet-121, DenseNet-169, DenseNet-201
Inception-based	Inception networks, also known as GoogleNet, use convolutional layers with different kernel sizes in parallel. This allows them to capture features at multiple scales and resolutions within the same layer. They are known for their effectiveness in reducing computation while improving performance.	InceptionV1, InceptionV2 (Inception-ResNet), InceptionV3, InceptionV4, Inception-ResNetV2

Category	Explanation	Examples
Xception-based	Xception, short for "Extreme Inception," is a deep learning architecture that builds on the ideas of the Inception network. Xception uses depthwise separable convolutions, which factorize standard convolutions into depthwise convolutions and pointwise convolutions. This reduces the number of parameters and computational cost while maintaining high accuracy.	Xception
SqueezeNet-based	SqueezeNet is a lightweight convolutional neural network architecture designed for low latency and small model size. It focuses on reducing the number of parameters and computational load while maintaining competitive accuracy.	SqueezeNet
NASNet-based	Neural Architecture Search (NAS) is an automated process that explores different network architectures to find the best-performing models. NASNet-based architectures are the result of this process and are designed to be highly efficient.	NASNet-A, NASNet-B

Factor 3: Training Strategy

Category	Explanation	Examples
Supervised	In supervised training, object detection models are trained on labeled datasets with ground-truth bounding boxes and class labels for each object. It is the most common training strategy.	Standard object detection with labeled bounding boxes.
Weakly Supervised	Weakly supervised object detection uses images with weak or noisy labels, such as image-level tags, instead of precise bounding box annotations.	Object localization using image-level tags.
Semi-Supervised	Semi-supervised approaches combine a small amount of labeled data with a large amount of unlabeled data. This helps in training when obtaining large quantities of labeled data is challenging.	Self-training, consistency regularization.

Category	Explanation	Examples
Few-Shot	Few-shot object detection aims to detect objects with very few examples, often just a handful of labeled instances. Few-shot learning techniques are employed to achieve this.	Few-shot learning techniques.

Factor 4: Classical Detection Techniques

Category	Explanation	Examples
Template Matching	Detecting objects by comparing image regions to predefined templates.	Cross-correlation, Normalized Cross-correlation (NCC), Template Matching with Scale and Rotation Invariance
Cascade Classifiers	Using a series of progressively more complex classifiers to detect objects.	Viola-Jones Face Detection, Haar Cascade Classifiers, LBP (Local Binary Pattern) Cascade
Feature-Based	Detecting objects by identifying distinctive features in the image.	Harris Corner Detector, Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), ORB (Oriented FAST and Rotated BRIEF)
Histogram-Based	Detecting objects by analyzing color or intensity histograms.	Histogram of Oriented Gradients (HOG), Color Histograms, Local Color Histograms
Edge Detection	Detecting objects by identifying edges and contours in the image.	Canny Edge Detector, Sobel Operator, Prewitt Operator, Zero Crossing Edge Detection
Corner Detection	Detecting objects by locating key corners or points in the image.	Harris Corner Detector, Shi-Tomasi Corner Detector, FAST (Features from Accelerated Segment Test)
Contour-Based	Detecting objects based on their contours and shapes.	Contour matching techniques, Polygonal Approximation, Shape Context
Blob Detection	Detecting objects as regions of interest with specific characteristics.	Blob detection using Laplacian of Gaussian (LoG), MSER (Maximally Stable Extremal Regions), Circular Hough Transform
Grayscale Images	Object detection in grayscale images.	Grayscale image-based object detection techniques.

Category	Explanation	Examples
Color Images	Object detection in color images.	Color image-based object detection methods.
Multi-Channel Data	Object detection in multi-channel data (e.g., color, depth).	Object detection methods using data with multiple channels.
Infrared Images	Object detection in infrared images.	Infrared image-based object detection for night vision and thermal imaging.
Depth Images	Object detection using depth information.	Kinect depth images for 3D object detection.
Lidar Data	Object detection using Lidar point cloud data.	Lidar-based 3D object detection for autonomous vehicles.
Multispectral Images	Object detection in images with multiple spectral bands.	Multispectral image-based object detection for remote sensing.

Each of these factors and categories plays a crucial role in defining the characteristics and capabilities of object detection models. The choice of which factors to prioritize depends on the specific requirements of the application in which the object detection system will be used.

1.11 Conclusion

In Conclusion, this chapter offered an overview of AI-driven solutions in the field of computer vision and object detection. We examined the drawbacks of conventional approaches and underscored the potential advantages of AI. We explored the capabilities of the human visual system and the important role of deep learning in crafting AI solutions.

Moreover, we introduced the most important models used in object detection based deep learning approaches, key benchmark datasets and the widely-used frameworks in deep learning. This chapter serves as the cornerstone for forthcoming chapters, which will delve into the Creation of our proposed model and the used methodologies for the detection of the desired object with the minimum cost time/memory and the highest accuracy and precision.

Chapter 2

The Convolutional neural networks (CNNs) in Depth

2.1 Introduction

In Deep Learning, Convolutional neural network (CNN or ConvNet) is a type of feed-forward artificial neural network in which the connectivity pattern between neuron is inspired by the organization of the human visual cortex.

In this section, we will explore Convolutional Neural Networks (CNNs) and their main components, processes, and techniques. CNNs are a vital part of deep learning and have revolutionized computer vision. We will delve into the mathematical formulations for each step and discuss normalization, activation functions, stride, and regularization, as well as output functions for different tasks. We study the basic principles of CNN:

- The most important methods built into CNNs, such as convolution and pooling...
- The different layers presented in CNN architecture, from the input to the fully connected and output layers, with detailed definitions including data types and the functions applied to them.
- Techniques used to improve CNN performance, such as normalization, optimization, and regularization, and how to avoid overfitting or underfitting.

2.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs), are a class of deep learning models that have revolutionized image analysis and object detection tasks. They are a fundamental component in the field of computer vision and play a critical role in object detection and comprehension in sequences of images. They have gained immense popularity due to their ability to automatically learn and extract meaningful features from raw data.

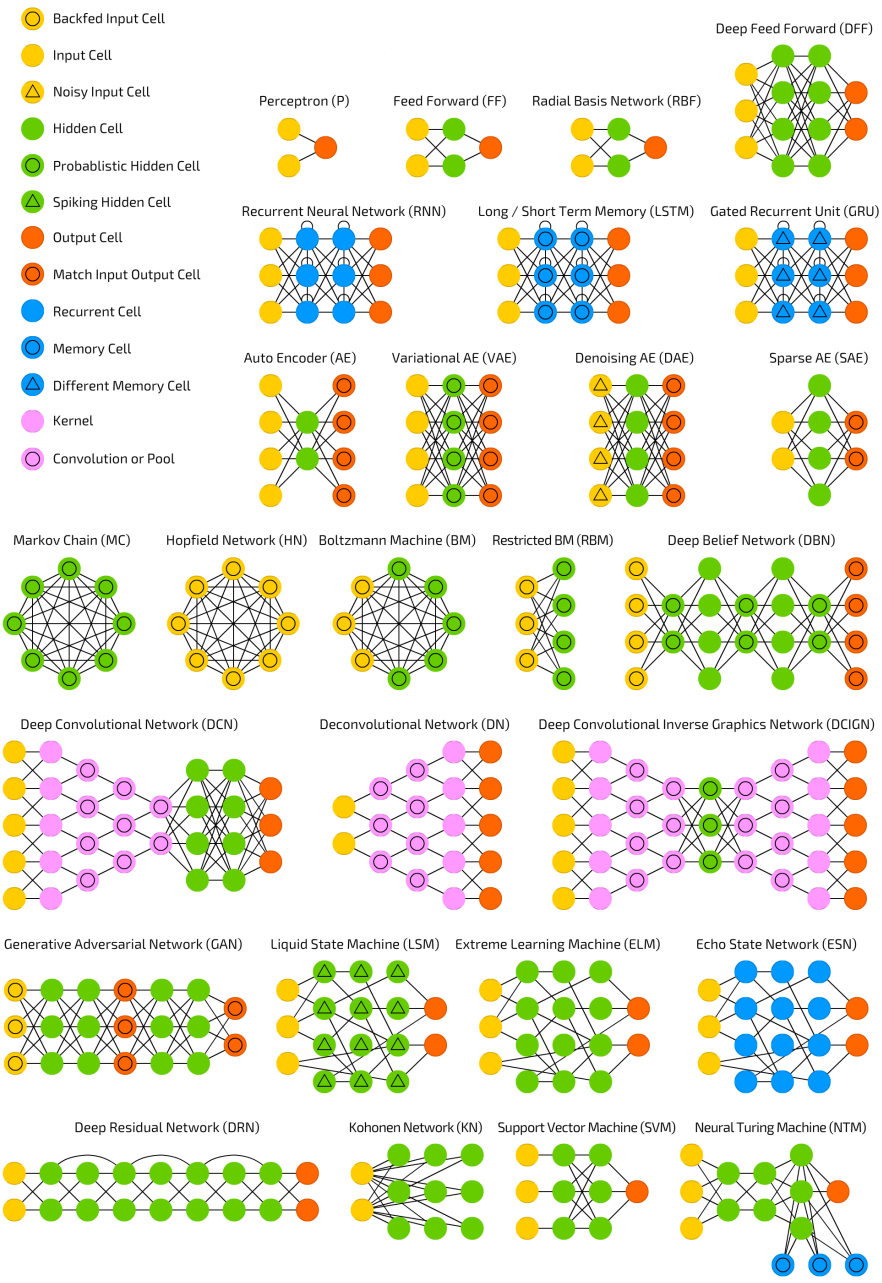


Figure 2.1: Different Architectures of deep learning

2.3 Why CNN for Computer Vision

The adult human primary visual cortex in each hemisphere is estimated to contain approximately 140 million neurons on average. Within this cortex, there are small regions of cells that exhibit sensitivity to specific areas of the visual field. Certain individual neuronal cells within the brain respond, or "fire," only when exposed to edges of particular orientations. For instance, some neurons are activated by vertical edges, while others are triggered by horizontal or diagonal edges.

Firstly, computers perceive images as collections of color intensities known as pixels. In the case of a colored RGB image, each pixel comprises three values representing Red, Green, and Blue. Consequently, the image is represented as a matrix with three dimensions: Width,

Height, and Depth (where depth equals 3 in RGB images).

Secondly, when it comes to image classification tasks, convolutional neural networks (CNNs) are preferred over traditional neural networks such as multi layer perceptrons (MLPs) [54]. In CNNs, neurons in a layer are only connected to a small region of the preceding layer, unlike in MLPs where neurons are part of fully connected networks. The drawbacks of MLPs include:

- MLP use one perceptron (neuron) for each input (e.g. pixel in an image, multiplied by 3 in RGB case). The amount of weights rapidly becomes unmanageable for large images. For a 224 x 224 pixel image with 3 color channels there are around 150,000 weights that must be trained! As a result, difficulties arise whilst training and over-fitting can occur.
- MLPs react differently to an input (images) and its shifted version, they are not translation invariant. For example, if a picture of a cat appears in the top left of the image in one picture and the bottom right of another picture, the MLP will try to correct itself and assume that a cat will always appear in this section of the image (***figure 2.2***).

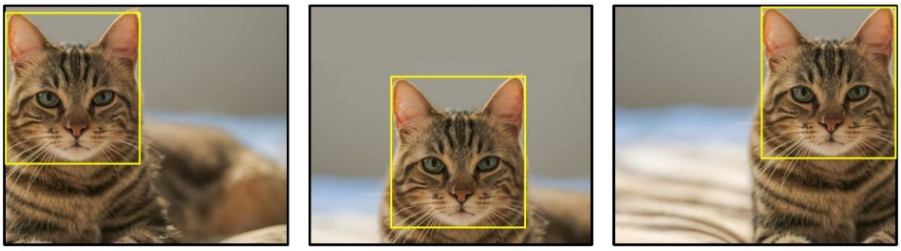


Figure 2.2: detecting objects

2.3.1 Key Components of CNNs

The basic architecture of CNNs consists of several layers, as illustrated in Figure 2.3. The primary distinction among various CNN implementations lies in the selection, arrangement, and parameterization of these layers and methods.

In this section, we will explore the fundamental building blocks and components of CNNs, which include convolutional layers, pooling layers, and fully connected layers.

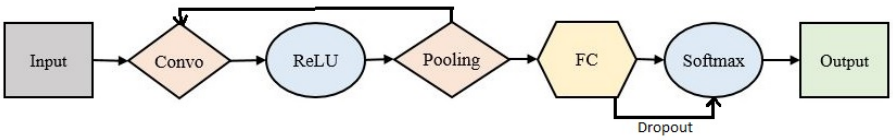


Figure 2.3: CNN general architecture

a more detailed CNN architecture can be seen in Figure 2.4.

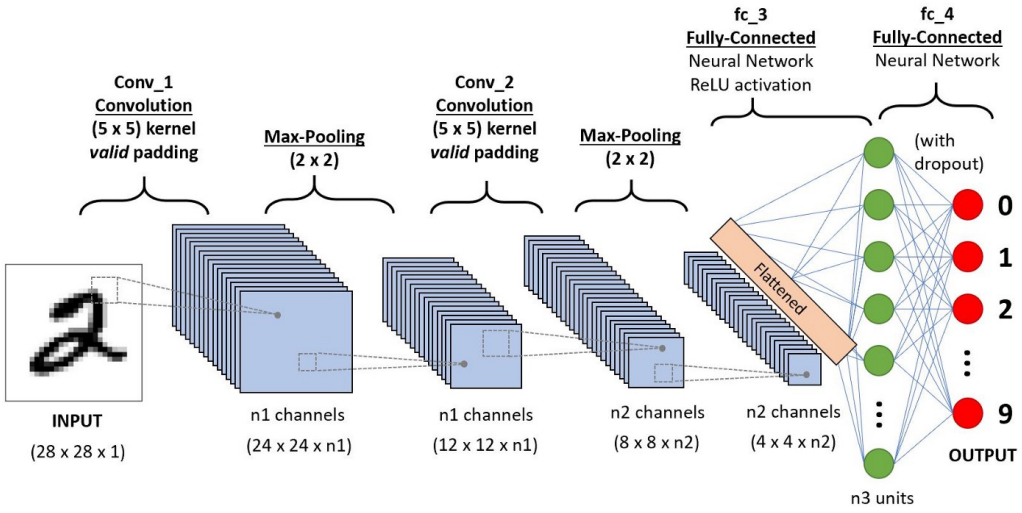


Figure 2.4: Common CNN architecture

2.3.1.1 Input Layer:

In a Convolutional Neural Network (CNN), the input layer holds the raw pixel values of the image like the input image in Figure 2.4. Image data is typically represented by a 3D matrix, with dimensions corresponding to width, height, and depth. For example, in the case of a colored RGB image, the depth would be 3 (representing Red, Green, and Blue channels).

In contrast, in a traditional multi layer perceptron (MLP) [54], the image needs to be reshaped into a single column vector before being fed into the input layer. For instance, if an image has dimensions of 28 x 28 pixels, it would be reshaped into a vector of size 784 x 1. If there are "m" training examples, the dimension of the input would then be (784, m), where "m" represents the number of training examples.

However, in CNNs, the image data remains in its original shape, i.e., as a matrix, preserving the spatial information of the image. This allows CNNs to effectively capture and utilize the local relationships between pixels through operations like convolution and pooling.

2.3.1.2 Convolutional Layers:

CNNs utilize convolutional layers [55] to apply a series of learnable filters (kernels) to the input data. These filters convolve over the image, enabling the network to detect features and patterns, generating what we call "feature maps". The convolution operation is defined as:

$$(f * g)(x, y) = \sum_i \sum_j f(i, j) \cdot g(x - i, y - j) \tag{2.1}$$

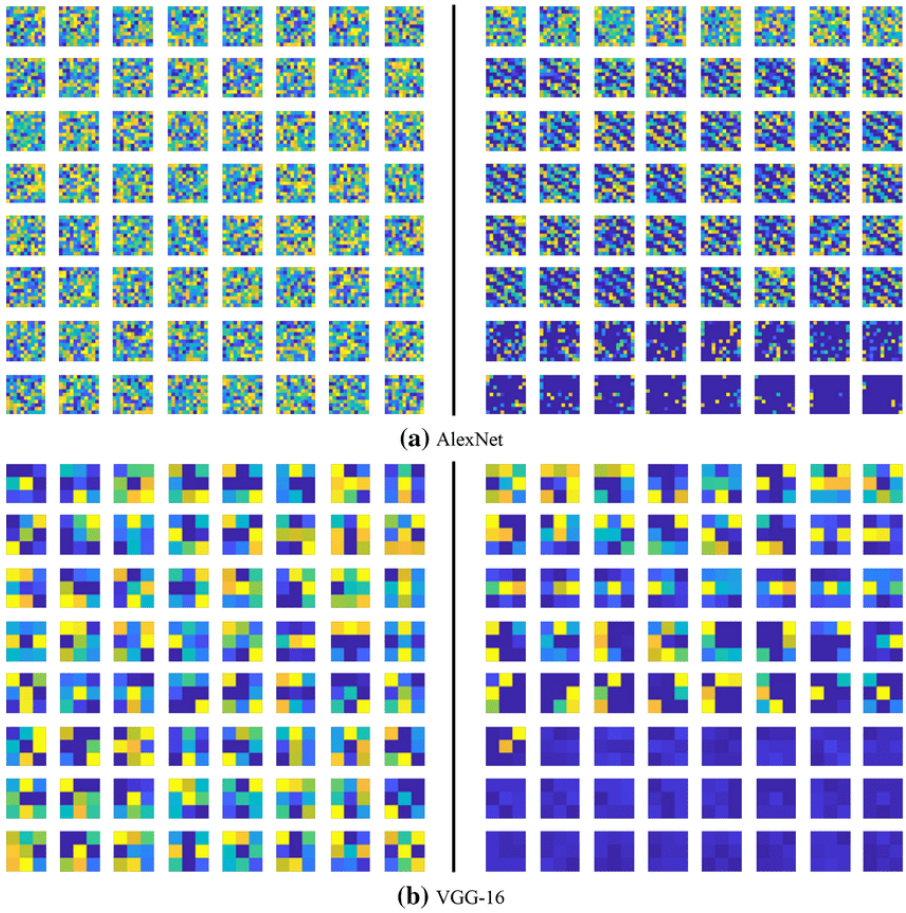


Figure 2.5: Visualization of filters in the first convolutional layer of (a) AlexNet and (b) VGG-16 for ImageNet.

where in equation (2.1) f is the image and g is the filter applied at each spatial coordinates x and y, with i and j are the filter indices. And multiple filters are applied in parallel, generating different feature maps that capture various aspects of the data.

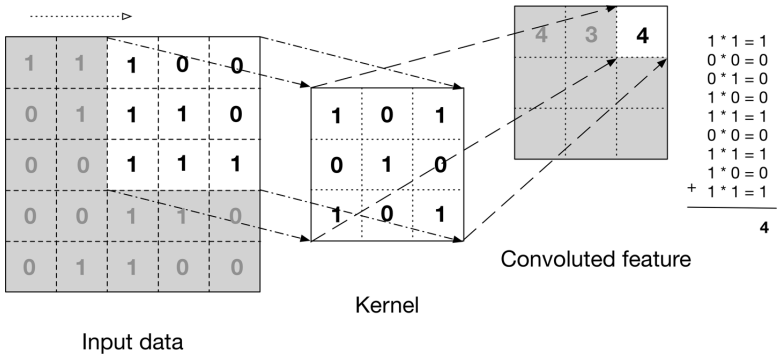


Figure 2.6: Convolutional layer.

Convolution with Stride and Padding

In practice, convolutional layers often use strides and padding to control the output size. Stride determines how much the filter moves between applications, and padding adds additional values to the input to control the spatial dimensions of the output feature map.

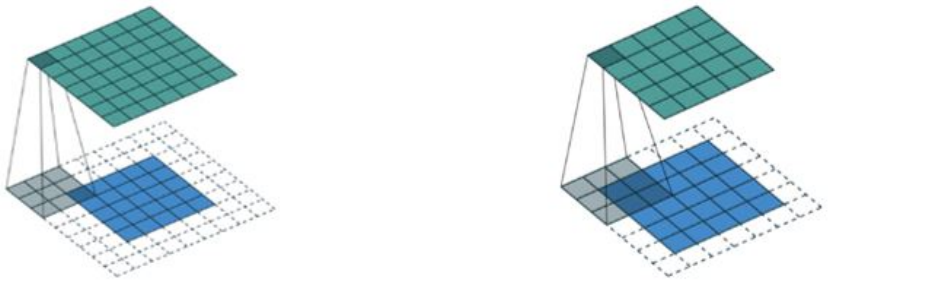
$$O = \frac{(I - F + 2P)}{S} + 1 \tag{2.2}$$

Where:

- O is the output size.
- I is the input size.
- F is the filter size.
- P is the padding.
- S is the stride.

To handle border effects and control the spatial dimensions of the output feature maps like in figure 2.8, ***padding*** is often applied. Padding can be "valid" (no padding) or "same" (zero-padding to keep the output size the same as the input size). The choice of padding impacts the feature map dimensions and, in turn, the network's ability to capture information near the image borders.

Padding Applying convolutions on a normal image reduce the size of the image by an amount depending on the size of the filter. Hence why padding is used to eliminate the loss of what can be an important part of the image.



Full padding: Introduces zeos such that all pixels are visited the same amount of time by the filter, resulting an increase in the size of the output

Same padding: Ensures that the output has the same size as the input

Figure 2.7: Padding

After convolution, an activation function is typically applied, commonly using Rectified Linear Unit (ReLU) [56]:

$$\text{ReLU}(x) = \max(0, x) \tag{2.3}$$

Stride

The stride is a hyperparameter that determines how much the convolutional kernel shifts during the convolution operation. A stride of 1 means that the kernel moves one pixel at a time, while a larger stride skips pixels and reduces the spatial dimensions of the output feature map.

The output size of a convolution operation with stride S can be represented as:

$$O = \frac{(I - F)}{S} + 1 \tag{2.4}$$

- Strided convolutions can be used to control the spatial resolution of feature maps and reduce computational complexity in deep networks.

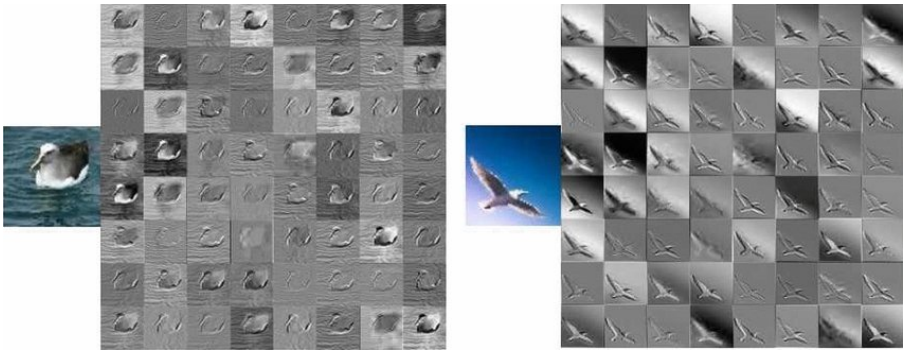


Figure 2.8: Visualize the feature map from the first convolutional layer of VGG-16.

2.3.1.3 Pooling Layers:

Pooling layers [57] are used to pool features together, often downsample the feature maps produced by convolutional layers to a smaller spatial dimensions and computational complexity. They can also induce favourable properties such as translation invariance in image classification, as well as bring together information from different parts of a network by using different scales of pooling.

A common pooling operation is **max-pooling**, where the maximum value in a local window is selected.

- Max-Pooling Max-Pooling [58] extracts the maximum value within a local window or region of the input feature map. The idea is to retain the most prominent feature within that region while discarding less important information. The typical window size for max-pooling is 2x2 or 3x3, and the window is moved across the input feature map with a certain stride. The maximum value within each window becomes the value in the downsampled feature map.

For a window size of $n \times m$ and a stride of s:

$$\text{Max-Pooling}(x, y) = \max_{i=0}^{n-1} \max_{j=0}^{m-1} I(x + i \cdot s, y + j \cdot s) \tag{2.5}$$

Where:

- x and y represent the spatial coordinates.
- i and j are the indices within the pooling window.
- I is the input feature map.

Benefits of Max-Pooling:

- Pooling layers help retain essential features while discarding less important information.
- Reducing spatial dimensions makes the network more computationally efficient.

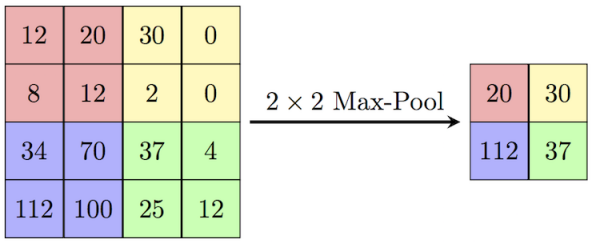


Figure 2.9: maxpooling

Other Pooling methods can be choosen:

- Average-Pooling Average-Pooling is another common pooling operation that computes the average value within a local window or region of the input feature map. Like max-pooling, average-pooling typically uses a 2x2 or 3x3 window and a certain stride to move across the input. The average value within each window becomes the value in the downsampled feature map.

Average-Pooling computes the average value within a local window or region of the input feature map:

For a window size of $n \times m$ and a stride of s:

$$\text{Average-Pooling}(x, y) = \frac{1}{n \cdot m} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} I(x + i \cdot s, y + j \cdot s) \tag{2.6}$$

Where:

- x and y represent the spatial coordinates.
- i and j are the indices within the pooling window.
- I is the input feature map.

Benefits of Average-Pooling:

- It helps reduce spatial dimensions and computational complexity.
- It provides a smoother downsampled feature map compared to max-pooling, which can be useful in certain applications.

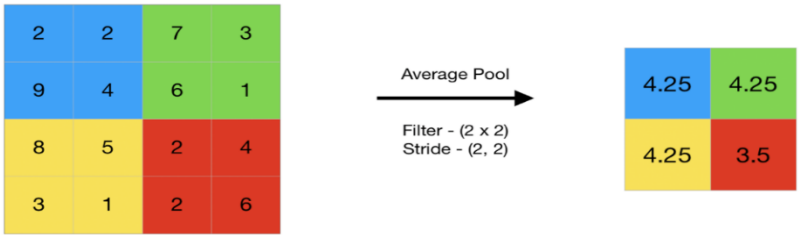


Figure 2.10: average pooling

- Global Average Pooling Global Average Pooling (GAP) [59] is a special type of pooling operation where the entire feature map is pooled into a single value. Unlike max-pooling and

average-pooling, which operate on local windows, GAP computes the average of all values across the entire feature map. This results in a fixed-size output, regardless of the size of the input feature map.

Global Average Pooling (GAP) computes the average value over the entire feature map:

$$\text{GAP} = \frac{1}{H \cdot W} \sum_{x=0}^{H-1} \sum_{y=0}^{W-1} I(x, y) \tag{2.7}$$

Where:

- H and W represent the height and width of the feature map, respectively.
- I is the input feature map.

Benefits of Global Average Pooling:

- GAP is often used in the final layers of CNN architectures for image classification. It helps reduce the number of parameters in the network.
- It forces the network to focus on the most critical features for the classification task.
- It introduces global spatial context, which can be useful for certain tasks.

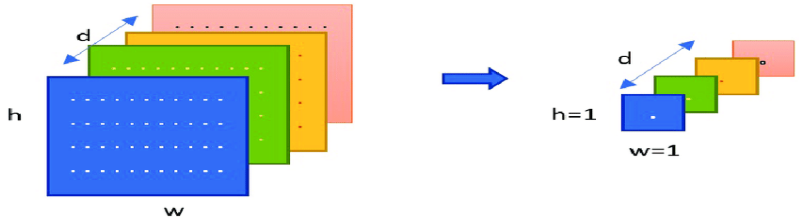


Figure 2.11: Globale average polling

Max-pooling retains the most prominent feature within a local window, average-pooling calculates the average value within a window, and global average pooling computes the average over the entire feature map. These pooling operations are important for reducing the spatial dimensions of feature maps, improving computational efficiency, and promoting translation invariance, depending on the specific needs of the task at hand. The choice of pooling operation and its parameters can significantly impact the performance of a CNN in various computer vision applications.

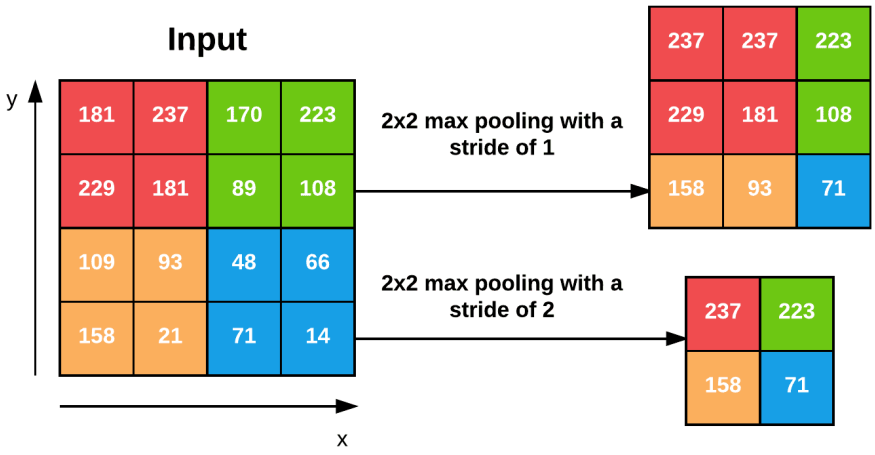


Figure 2.12: Max Pooling layer 2x2 with stride (of 2) or without stride (of 1).

In practice, we tend to see two types of max pooling variations:

- **Type 1:** F = 3, S = 2, which is called overlapping pooling and normally applied to images/input volumes with large spatial dimensions.
- **Type 2:** F = 2, S = 2, which is called non-overlapping pooling. This is the most common type of pooling and is applied to images with smaller spatial dimensions.

For CNN architectures that accept smaller input images (32 to 64 pixels) we may see F = 2, S = 1.

other used Pooling methods are:

Spatial Pyramid Pooling Introduced by He et al. [60] in Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. Spatial Pyramid Pooling (SPP) is a pooling layer that removes the fixed-size constraint of the network, i.e. a CNN does not require a fixed-size input image. Specifically, we add an SPP layer on top of the last convolutional layer. The SPP layer pools the features and generates fixed-length outputs, which are then fed into the fully-connected layers (or other classifiers). In other words, we perform some information aggregation at a deeper stage of the network hierarchy (between convolutional layers and fully-connected layers) to avoid the need for cropping or warping at the beginning.

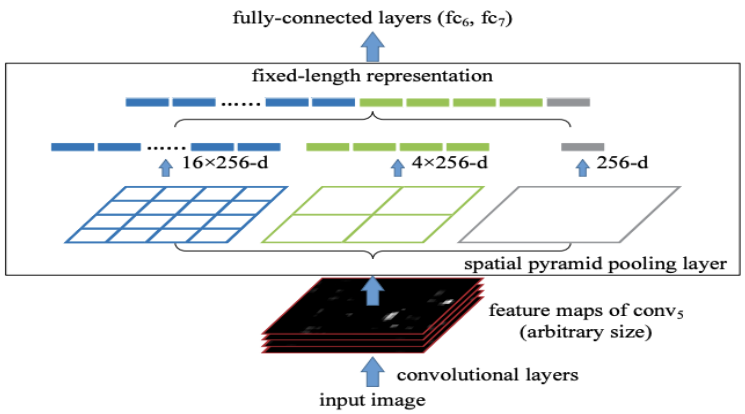


Figure 2.13: Spatial Pyramid Pooling (SPP).

Center Pooling (Introduced by Duan et al. in CenterNet: Keypoint Triplets for Object Detection [61]) is a pooling technique for object detection that aims to capture richer and more recognizable visual patterns. The geometric centers of objects do not necessarily convey very recognizable visual patterns (e.g., the human head contains strong visual patterns, but the center keypoint is often in the middle of the human body).

The detailed process of center pooling is as follows: the backbone outputs a feature map, and to determine if a pixel in the feature map is a center keypoint, we need to find the maximum value in its both horizontal and vertical directions and add them together. By doing this, center pooling helps the better detection of center keypoints.

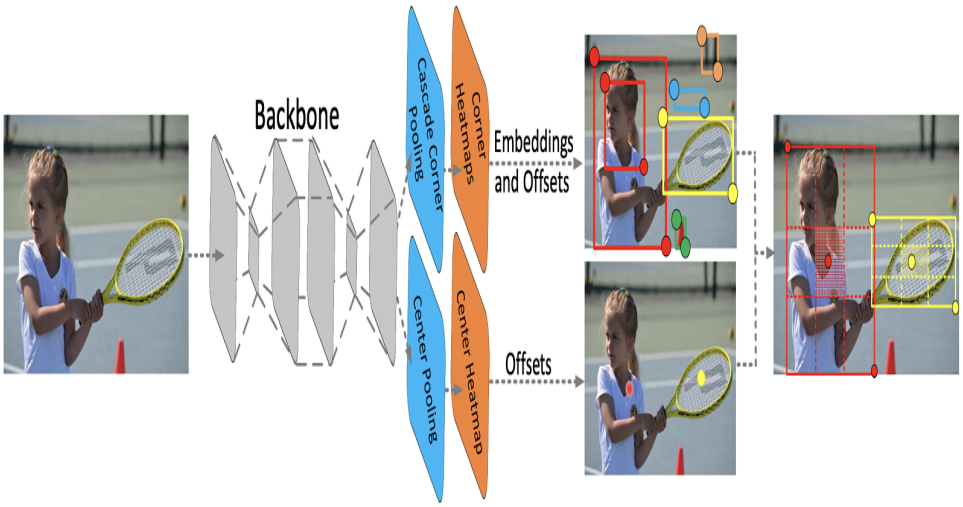


Figure 2.14: Center Pooling.

Cascade Corner Pooling Also introduced by Duan et al. [61], is a pooling layer for object detection that builds upon the corner pooling operation. Corners are often outside the objects, which lacks local appearance features. CornerNet uses corner pooling to address this issue, where we find the maximum values on the boundary directions so as to determine corners. However, it makes corners sensitive to the edges. To address this problem, we need to let corners see the visual patterns of objects. Cascade corner pooling first looks along a boundary to find a boundary maximum value, then looks inside along the location of the boundary maximum value to find an internal maximum value, and finally, add the two maximum values together. By doing this, the corners obtain both the the boundary information and the visual patterns of objects.

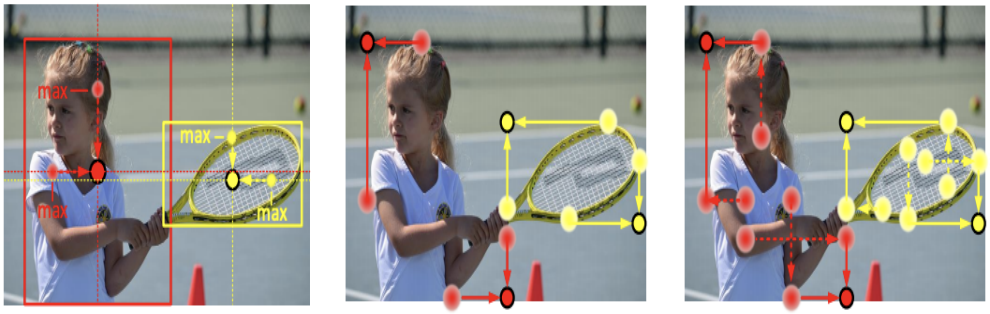


Figure 2.15: Cascade Corner Pooling.

in the following figure 2.16, we can see the most used Pooling Operations in the last years.

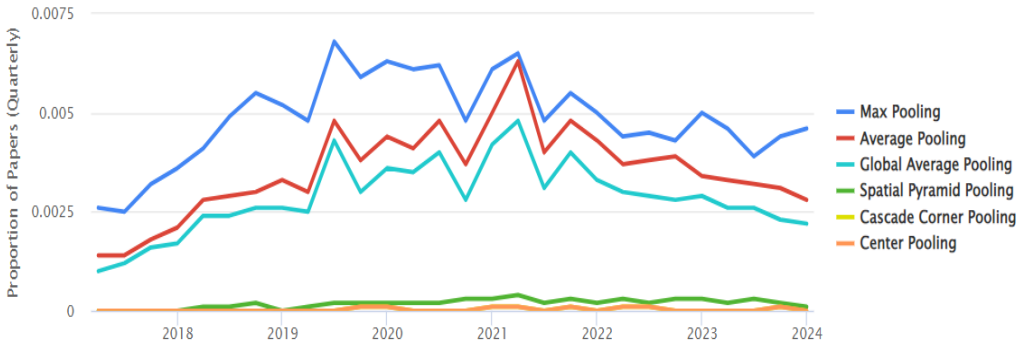


Figure 2.16: Most used Pooling Operations in the last years.

2.3.1.4 Dense or Fully Connected Layers:

A fully connected layer connects every neuron to every neuron in the previous and subsequent layers, effectively creating a dense matrix of connections [62]. After a series of convolutional and pooling layers, CNNs typically employ one or more fully connected layers for high-level feature abstraction. These layers are similar to traditional feedforward neural networks, and they play a critical role in making final predictions or decisions. The output of a fully connected layer is often computed as:

$$\text{Output} = \sigma(Wx + b) \tag{2.8}$$

Where:

- $Output$ is the output, and can be $Output(i)$ the output of every neuron i.

- σ represents the activation function applied to the layer's output, commonly using the ReLU function, sigmoid, or hyperbolic tangent (tanh).

- W is the weight matrix, where each weight $Weight(j, i)$ denotes the connection strength between the j-th neuron in the current layer and the i-th neuron in the next layer.

- x is the input vector, typically the output from the previous layer $Input(j)$.

- b is the bias vector, which allows the layer to introduce an offset "$Bias(i)$".

the formulas can be rewritten as:

$$\text{Output}(i) = \sigma\left(\sum_{j=1}^{N} \text{Input}(j) \cdot \text{Weight}(j,i) + \text{Bias}(i)\right) \tag{2.9}$$

Fully connected layers are often employed in the later stages of a CNN for tasks such as classification, where high-level features are combined and processed to make predictions. They play a crucial role in transforming the extracted features into meaningful output, making them a key component in the architecture of CNNs.

Role of Dense Layers

Dense layers are responsible for connecting every neuron in one layer to every neuron in the next layer. They serve several essential purposes in neural networks:

- *Learning Complex Patterns:* Dense layers learn and model complex patterns and relationships in the data by forming combinations of features from the previous layer.
- *Feature Extraction:* They transform the input data into a higher-level representation that is increasingly abstract and meaningful for the given task.
- *Non-Linearity Introduction:* Dense layers introduce non-linearity into the network through the activation functions applied to the output of each neuron.

Common Use Cases

Dense layers are commonly used in various types of neural network architectures, including:

- *Feedforward Neural Networks (FNNs):* In FNNs, dense layers are stacked together to form the hidden layers of the network. They play a vital role in learning complex mappings between inputs and outputs.
- *Deep Neural Networks (DNNs):* DNNs often consist of multiple dense layers, allowing them to model highly intricate patterns and relationships in data.
- Autoencoders: Dense layers are used in both the encoding and decoding parts of autoencoders, which are unsupervised learning models for feature extraction and data compression.

2.3.2 Weights:

Among the key components of Convolutional Neural Networks (CNNs), one of the most fundamental is the set of **weight parameters**. Weights are learnable parameters that define the strength of connections between neurons in the network. In the context of CNNs, these weights determine how information flows from one layer to another and are essential for feature extraction and decision making.

Weights in Convolutional Layers: In convolutional layers, weights are applied to a small region of the input data, known as the receptive field or filter. These weights act as filters that

slide across the input image, extracting features by performing element-wise multiplications and summing the results. The learned weights enable the network to detect specific patterns, edges, textures, or more complex features within the input data. Mathematically, the convolution operation for a single feature map can be defined as:

$$(F * W)(x, y) = \sum_i \sum_j F(i, j) \cdot W(x - i, y - j) \tag{2.10}$$

Where: - F is the input feature map or image. - W is the weight filter.

Weights in Fully Connected Layers: Weights play a significant role in fully connected layers, where each neuron in the layer is connected to every neuron in the previous and subsequent layers. These weights determine the strength of connections and influence the high-level feature abstraction and decision-making process. Learning the optimal weights is essential for the network to make accurate predictions based on the extracted features. Mathematically, the output of a single neuron in a fully connected layer can be defined as:

$$\text{Output}_j = \sigma\left(\sum_i W_{ji} \cdot \text{Input}_i + b_j\right) \tag{2.11}$$

Where: - W_{ji} is the weight connecting neuron i in the previous layer to neuron j in the fully connected layer. - σ is the activation function, such as ReLU or sigmoid. - b_j is the bias term for neuron j.

Weights in Training Process: Weights in CNNs are initially assigned random values and are iteratively adjusted during the training process to minimize a chosen loss function. The goal is to find the weights that allow the network to make accurate predictions by reducing the difference between predicted and ground truth values.

Weights in Transfer Learning: Transfer learning often involves reusing pre-trained CNN models on large datasets. These pre-trained models have already learned valuable feature representations, and the weights encapsulate this knowledge. Fine-tuning these weights on a specific task can significantly speed up the training process and improve the network's performance.

2.3.3 Normalization and Activation Functions

Normalization and activation functions are essential components in CNNs to enhance training stability and feature extraction.

2.3.3.1 Normalization

Normalization techniques, like Batch Normalization (BatchNorm) [63], are applied to ensure that the network's inputs have a similar scale, which can help in the training process. BatchNorm normalizes the output of each layer within a mini-batch, effectively stabilizing and accelerating training.

The Mathematical Formulation:

$$\text{BatchNorm}(x) = \gamma \cdot \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{2.12}$$

- γ and β are learnable scale and shift parameters.
- μ and σ^2 are the mean and variance of the mini-batch.
- ϵ is a small constant to prevent division by zero.

Normalization techniques are essential for preprocessing data in neural networks to ensure stable and efficient training. Here are some commonly used normalization techniques:

Standardization (Z-score normalization):

$$x_{\text{standardized}} = \frac{x - \mu}{\sigma} \tag{2.13}$$

Where:

- x is the original feature.
- μ is the mean of the feature across the dataset.
- σ is the standard deviation of the feature across the dataset.

Min-Max Scaling:

$$x_{\text{scaled}} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{2.14}$$

Where:

- x is the original feature.
- $\min(x)$ is the minimum value of the feature across the dataset.
- $\max(x)$ is the maximum value of the feature across the dataset.

Batch Normalization:

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{2.15}$$

$$y = \gamma \hat{x} + \beta \tag{2.16}$$

Where:

- x is the input to the layer.
- μ_B is the mean of x over the current mini-batch.
- σ_B is the standard deviation of x over the current mini-batch.
- ϵ is a small constant added to avoid division by zero.
- γ and β are learnable parameters (scale and shift, respectively).

Layer Normalization:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.17}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{2.18}$$

$$y = \gamma\frac{x - \mu}{\sigma} + \beta \tag{2.19}$$

Where:

- N is the number of neurons in the layer.
- x_i is the i-th input to the layer.
- μ and σ are the mean and standard deviation of the inputs across neurons.
- γ and β are learnable parameters (scale and shift, respectively).

2.3.3.2 Activation Functions

Activation functions are vital components of artificial neural networks, as they govern the output of individual neurons or entire layers of neurons. These functions introduce non-linearity to the network, which is essential for enabling it to learn complex relationships within the data. In this section, we will delve into the importance of activation functions and discuss some commonly used types.

2.3.3.3 Purpose of Activation Functions

In neural networks, activation functions serve several essential purposes:

- *Introduce Non-Linearity:* Activation functions enable neural networks to model and approximate non-linear relationships within data. Without them, the entire network would be a linear transformation.
- *Enable Complex Representations:* They allow networks to represent complex functions by stacking multiple layers and activation functions together, forming deep neural networks.
- *Thresholding:* They introduce a threshold for neuron activation. If the input is above a certain threshold, the neuron becomes active, enabling the network to make binary decisions.

2.3.3.4 Common Activation Functions

There are several types of activation functions used in neural networks. Here are some of the most commonly used ones:

0. **Rectified Linear Unit (ReLU):** The ReLU [56] activation is one of the most popular choices for its simplicity and effectiveness. It replaces all negative values with zero and is mathematically defined as:

$$\text{ReLU}(x) = \max(0, x) \tag{2.20}$$

0. **Sigmoid:** The Sigmoid [64] function is a classic activation function that maps its input to a range between 0 and 1. It's especially useful in binary classification problems and historically in feedforward neural networks.

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{2.21}$$

0. **Hyperbolic Tangent (tanh):** The tanh activation is similar to the sigmoid but maps values to the range (-1, 1). It is zero-centered, making it easier to train deep networks. It is defined as:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.22}$$

0. **Leaky ReLU:** Leaky ReLU is a variant of ReLU that allows a small gradient for negative values, preventing neurons from being "dead" during training. It is defined as:

$$\text{Leaky ReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases} \tag{2.23}$$

   Here, α is a small positive constant (usually 0.01).

0. **Parametric Rectified Linear Unit (PReLU)**

   PReLU is an extension of Leaky ReLU, where the slope for negative values is learned during training. It has the advantage of adaptively changing the slope.

$$\text{PReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ ax & \text{if } x < 0 \end{cases} \tag{2.24}$$

   Here, a is a learnable parameter.

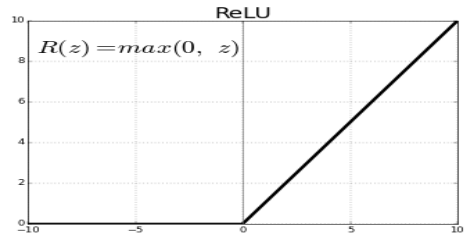
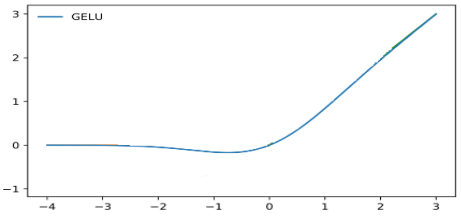
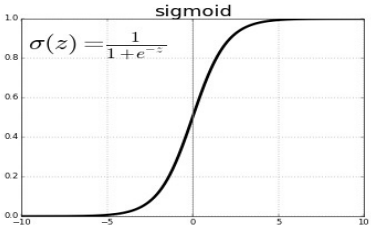
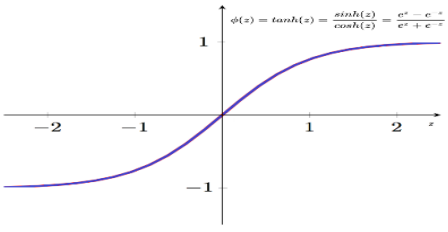
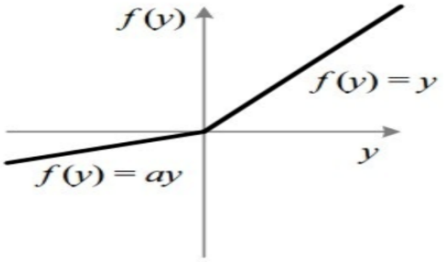
2.3.3.5 Choosing the Right Activation Function

The selection of an activation function is contingent upon the specific task at hand, and the architecture of the neural network can profoundly influence its performance. While Rectified Linear Unit (ReLU) and its variants are widely utilized as activation functions, there is no universally optimal choice. Experimentation and a comprehensive understanding of the distinctive

traits of different functions are pivotal in determining the most suitable activation function for a given problem.

To recapitulate, activation functions serve as pivotal elements in neural networks by introducing non-linearity, thereby facilitating the modeling of intricate relationships within data. The choice of activation function can substantially affect the network's efficacy, underscoring the importance of comprehending their characteristics in constructing proficient neural network architectures.

Table 2.1 shows a recapitulate of the Ten most used Activation functions.

Function	**graph**	**Formula**	**year**
ReLU		f (x) = max(0, x)	2000
GELU		$f(x) = x.\frac{1}{2}\left[1 + erf\left(\frac{x}{\sqrt{2}}\right)\right]$	2016
Sigmoid Activation		$f(x) = \left(\frac{1}{1+\ e^{-x}}\right)$	2000
Tanh Activation		$f(x) = \left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right)$	2000
Leaky ReLU		f(x)= α x +x = $\begin{cases} x\ si & x > 0 \\ ax\ si & x \leq 0 \end{cases}$	2014

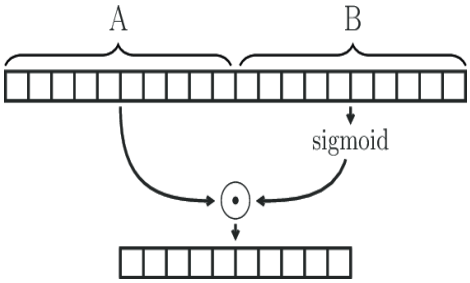
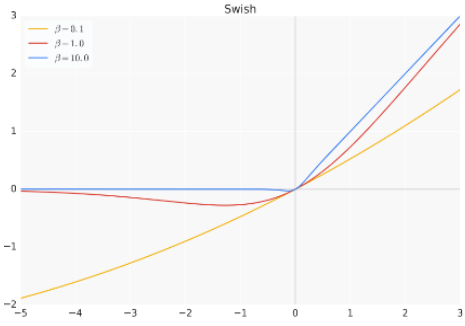
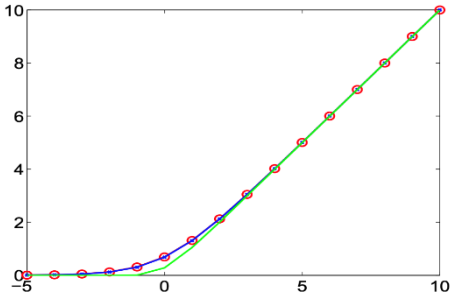
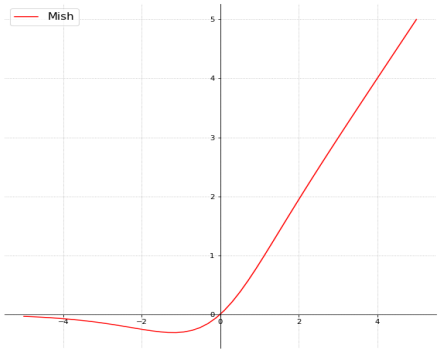
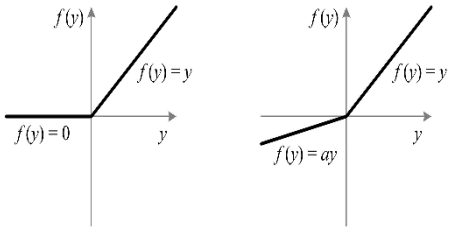
Function	graph	Formula	year
GLU			2016
Swish		$f(x) = x.sigmoid(x)$	2017
Softplus		$f(x)$=$ln(1+ \exp^{x})$	2000
Mish		$F(x) = Tanh\ Softplus(x)$	2019
PReLU		$f(x) = \begin{cases} x_i\ , si\ x_i > 0 \\ a_i x_i\ ,\ si\ x_i \leq 0 \end{cases}$	2014

Tableau 2.1: top 10 of most used activation functions.

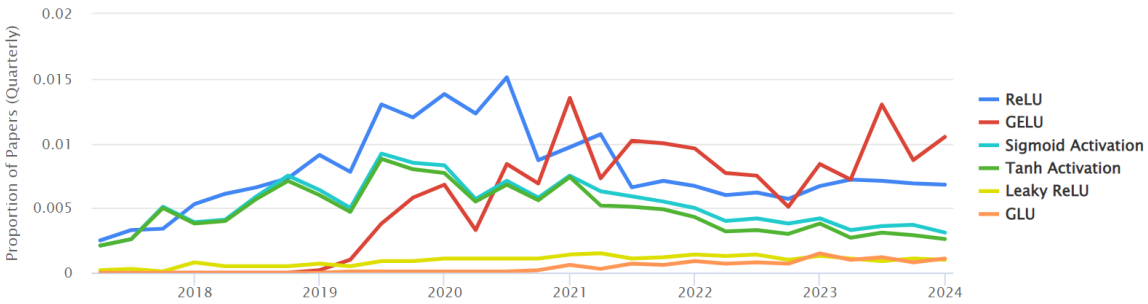


Figure 2.17: Best Activation methods time Line

2.4 Optimizers

Optimizers are algorithms or methods used to change the attributes of neural network such as weights and learning rate in order to reduce the losses. In the simplest case, an optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the value of the function. The generalization of optimization theory and techniques to other formulations constitutes a large area of applied mathematics. More generally, optimization includes finding "best available" values of some objective function given a defined domain (or input), including a variety of different types of objective functions and different types of domains.

2.4.0.1 Most known Optimizers:

Name	Advantages	Disadvantages
Gradient Descent	• Easy computation • Easy to implement • Easy to understand	• May trap at local minima • Weights are changed after calculating gradient on the whole dataset. So, if the dataset is too large than this may take years to converge to the minima. • Requires large memory to calculate gradient on the whole dataset.

Name	Advantages	Disadvantages
Stochastic Gradient Descent	• Frequent updates of model parameters hence, converges in less time. • Requires less memory as no need to store values of loss functions. • May get new minima's.	• High variance in model parameters. • May shoot even after achieving global minima. • To get the same convergence as gradient descent needs to slowly reduce the value of learning rate.
Mini-Batch Gradient Descent	• requently updates the model parameters and also has less variance. • Requires medium amount of memory.	• Mini-batch requires additional hyperparameters "mini-batch size" to be set for the learning algorithm. • Error information should be accumulated over a mini-batch of training samples, such as batch gradient descent.
Momentum	• Reduces the oscillations and high variance of the parameters. • Converges faster than gradient descent.	• One more hyper-parameter is added which needs to be selected manually and accurately.
Nesterov Accelerated Gradient	• Does not miss the local minima. • Slows if minima's are occurring.	• Still, the hyperparameter needs to be selected manually.
Adagrad	• Learning rate changes for each training parameter. • Don't need to manually tune the learning rate. • Able to train on sparse data.	• Computationally expensive as a need to calculate the second order derivative. • The learning rate is always decreasing results in slow training.
AdaDelta	• Now the learning rate does not decay and the training does not stop.	• Computationally expensive.
Adam	• The method is too fast and converges rapidly. • Rectifies vanishing learning rate, high variance.	• Computationally costly.

in the following figure 2.18, we can see the most used Optimization methods in the last

years.

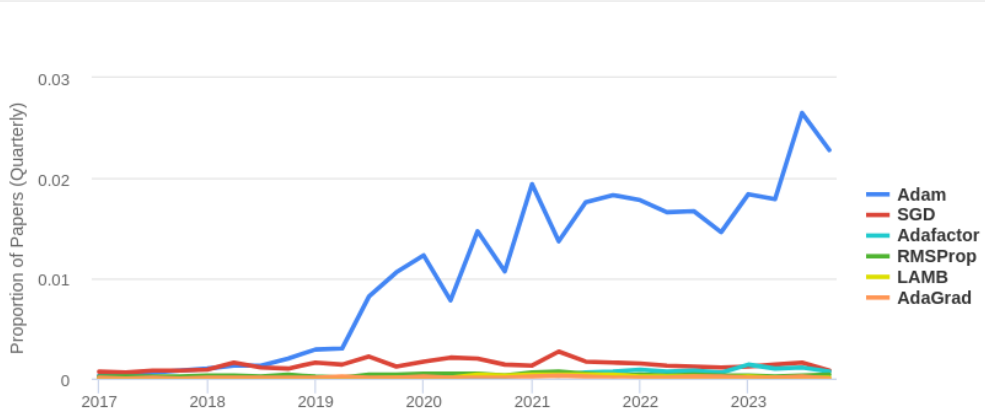


Figure 2.18: Optimization methods

2.4.0.2 Most Recent Optimization methods:

#	Name	Reference	Year
1	Lion	[65]	2023
2	SLAMB	[66]	2023
3	DeepZero	[67]	2023
4	Forward gradient	[68]	2023
5	AdaSmooth	[69]	2022
6	SRMM	[70]	2022

2.4.0.3 Most used Optimization methods:

#	Name	Reference	Year
1	Adam	Kingma D.P et al [71]	2014
2	SGD	Robbins H and Monro S [72]	1951
3	Adafactor	Shazeer M et al [73]	2018
4	RMSProp	Hinton G [74]	2013
5	AdaGrad	Rohan A et al [75]	2019
6	LAMB	You Y et al [76]	2020

2.5 Regularization Techniques in Neural Networks

In the field of machine learning and deep learning, regularization techniques are crucial for preventing overfitting and improving the generalization of neural networks. Overfitting occurs when a model learns the training data too well and performs poorly on unseen data. In this

section, we will explore various regularization methods used to combat overfitting and enhance the performance of neural networks.

Regularization strategies are designed to reduce the test error of a machine learning algorithm, possibly at the expense of training error. Many different forms of regularization exist in the field of deep learning.

2.5.1 The Need for Regularization

Neural networks are highly flexible models with a large number of parameters. While this flexibility allows them to fit complex patterns in data, it also makes them susceptible to overfitting, where they learn noise in the training data. Regularization methods help constrain the network's capacity, preventing it from memorizing the training data and encouraging it to capture meaningful patterns.

2.5.2 Common Regularization Techniques

Several regularization techniques are commonly used in neural networks:

2.5.2.1 Weight decay (L1 and L2 Regularization)

L1 Regularization

L1 regularization, also known as Lasso regularization, add a penalty term to the loss function during training. These penalties discourage large weights and promote sparsity (L1).

L1 regularization adds the absolute values of the weights to the loss function:

$$\text{Loss} = \text{Original Loss} + \lambda \sum_{i=1}^{n} |w_i| \tag{2.25}$$

Here, λ controls the strength of regularization.

L2 Regularization

Weight decay, L2 regularization, also known as Ridge regularization, , is a technique commonly used in machine learning and deep learning to prevent overfitting. it adds a penalty term to the loss function during training. These penalties prevent weights from becoming too large (L2).

L2 regularization adds the squares of the weights to the loss function:

$$\text{Loss} = \text{Original Loss} + \lambda \sum_{i=1}^{n} w_i^2 \tag{2.26}$$

Where: - Original Loss is the original loss function without regularization.

- λ is the regularization parameter, also known as the weight decay coefficient. It controls the strength of regularization. Higher values of λ impose stronger regularization, leading to smaller weights.

- n is the total number of weights in the model.

- w_i represents the ith weight in the model.

The regularization term $\lambda \sum_{i=1}^{n} w_i^2$ penalizes large weights because it grows quadratically with the magnitude of the weights. During training, the optimizer aims to minimize this combined loss, encouraging the model to learn simpler patterns and avoid over-fitting by keeping the weights small.

When the model updates its weights using gradient descent or a similar optimization algorithm, the gradient of the regularization term $\lambda \sum_{i=1}^{n} w_i^2$ is added to the gradients computed from the original loss function. This additional term nudges the optimizer to prefer weight values that are smaller in magnitude.

In practice, weight decay is a common technique used alongside other regularization methods like dropout to improve the generalization ability of neural networks. By penalizing large weights, weight decay helps prevent the model from fitting noise in the training data and encourages it to focus on learning the most important features.

2.5.2.2 Dropout

Dropout [77] is a popular regularization technique that randomly "drops out" a fraction of neurons during each training iteration. This prevents any single neuron from relying too heavily on specific features in the data, and to learn more robust and generalized features.

$$\text{Dropout}(x) = \frac{1}{1-p} \cdot x \tag{2.27}$$

- x is the input to a dropout layer. - p is the dropout probability, representing the fraction of neurons to deactivate.

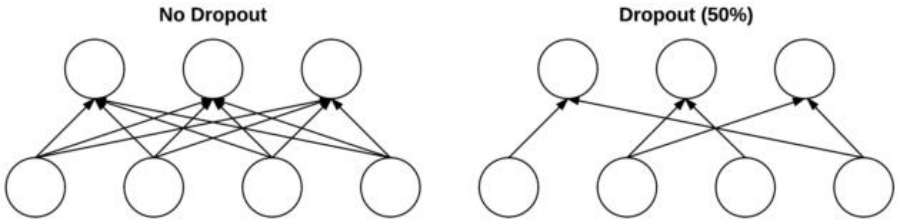


Figure 2.19: Left: Two layers of a neural network that are fully connected with no dropout. Right: The same two layers after dropping 50% of the connections.

It is a stochastic regularization technique that reduces over-fitting by (theoretically) combining many different neural network architectures. With Dropout, the training process essentially drops out neurons in a neural network. They are temporarily removed from the network, which can be visualized in 2.19.

This removal of neurons and synapses during training is performed at random, with a parameter p that is tunable (or, given empirical tests, best set to 0.5 for hidden layers and close to 1.0 for the input layer).

Dropout is a regularization technique for neural networks that drops a unit (along with connections) at training time with a specified probability p(a common value is $p = 0.5$). At test time, all units are present, but with weights scaled by p(i.e. w becomes pw).

The idea is to prevent co-adaptation, where the neural network becomes too reliant on particular connections, as this could be symptomatic of over-fitting. Intuitively, dropout can be thought of as creating an implicit ensemble of neural networks. The idea is to prevent co-adaptation, where the neural network becomes too reliant on particular connections, as this could be symptomatic of over-fitting.

2.5.2.3 Early Stopping

Early Stopping [78] is a regularization technique for deep neural networks that stops training when the validation performance starts to degrade, preventing the model from over-fitting

Early stopping involves monitoring the model's performance on a validation dataset during training.

In essence, we store and update the current best parameters during training, and when parameter updates no longer yield an improvement (after a set number of epochs/iterations) we stop training and use the last best parameters. It works as a regularizer by restricting the optimization procedure to a smaller volume of parameter space.

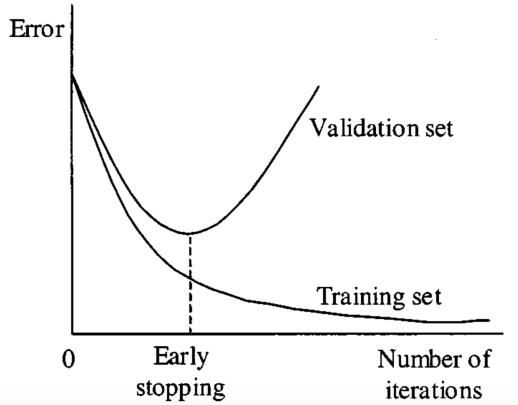


Figure 2.20: Early Stopping.

2.5.2.4 Data Augmentation

The common case in most machine learning applications, especially in image classification tasks where new training data is hard to obtain or expensive (such as medical resources). therefore, the creation of more data is an important step. We artificially boost the size of the training set, reducing over-fitting 2.21.

Data Augmentation for making simple alterations on visual data is popular. In addition, generative adversarial networks (GANs) are used to create new synthetic data.

Figure 2.21: Data augmentation in CNNs

Data augmentation involves creating new training examples by applying various transformations to the existing data, such as : padding, random rotating, re-scaling, vertical and horizontal flipping, translation (image is moved along X, Y direction), cropping, zooming, darkening & brighteningcolor, grayscaling, changing contrast, adding noise, random erasing...etc. data augmentation is only performed on the training data. The validation or test set remains unchanged.

Data augmentation is done dynamically during training time or before the training.

2.5.2.5 Most used Regularization Methods:

#	Reference	Name	Year
1	[77]	Dropout	2014
2	[79]	Label Smoothing	1985
3	[80]	Weight Decay	1943
4	[81]	Attention Dropout	2018
5	[82]	Entropy Regularization	2016
6	[83]	R1 Regularization	2018
7	[78]	Early Stopping	1995
8	[84]	Stochastic Depth	2016
9	[85]	Path Length Regularization	2019
9	[86]	Variational Dropout	2015

2.5.3 Choosing the Right Regularization Technique

The choice of regularization technique depends on the specific problem, dataset, and network architecture. Experimentation is often necessary to determine the most effective regularization strategy. Combining multiple regularization techniques can also be beneficial in many cases.

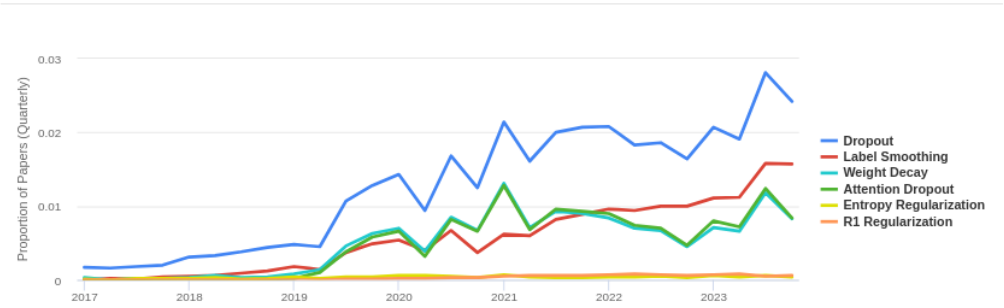


Figure 2.22: Best Regularization methods

2.6 Output Functions in Neural Networks

Output functions, also known as activation functions for the output layer, define how a neural network's final predictions or outputs are computed. The choice of output function depends on the nature of the problem and the desired output format, and it plays a critical role in determining the network's suitability for a given task. In this section, we will explore the significance of output functions and discuss common types used in neural networks.

2.6.1 The Role of Output Functions

The output layer of a neural network is responsible for producing the final predictions or outputs based on the information learned from the previous layers. The output function serves the following purposes:

- **Encoding Output Format**: It defines the format in which the network's predictions are expressed, whether it's continuous values for regression, probabilities for classification, or other custom formats.
- **Introducing Non-Linearity**: Output functions can introduce non-linearity in the predictions, which can be essential for modeling complex relationships in the data.
- **Matching Problem Requirements**: The choice of output function should align with the requirements of the specific problem, such as binary classification, multi-class classification, or regression.

2.6.2 Common Output Functions

There are several types of output functions used in neural networks, each designed for specific tasks. Here are some of the most commonly used ones:

2.6.2.1 Linear Activation Function

The Linear activation function is used for regression tasks, where the network is required to predict continuous values. It outputs the weighted sum of inputs without applying non-linearity.

$$\text{Linear}(x) = x \tag{2.28}$$

2.6.2.2 Sigmoid Function

The Sigmoid function is commonly used for binary classification tasks. It squashes the output into a range between 0 and 1, representing the probability of the input belonging to the positive class.

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{2.29}$$

2.6.2.3 Softmax Function

The Softmax output function transforms a previous layer's output into a vector of probabilities. It is commonly used for multiclass classification. It converts the network's raw scores into a probability distribution over multiple classes, ensuring that the predicted probabilities sum to 1. Given an input vector x and a weighting vector w we have:

$$\text{Softmax}(y = j \mid x) = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T wk}} \tag{2.30}$$

2.6.2.4 Rectified Linear Unit (ReLU)

In certain regression and classification tasks, ReLU can be used as an output function. While it's more commonly used as an activation function in hidden layers, it can serve as an output function for certain applications.

$$\text{ReLU}(x) = \max(0, x) \tag{2.31}$$

2.6.3 Choosing the Right Output Function

The choice of output function is determined by the nature of the task and the required output format. It is essential to align the output function with the problem's objectives, such as regression, binary classification, or multi-class classification. Careful consideration of the task's requirements is crucial in selecting the appropriate output function.

Output functions are a critical component of neural networks, defining how the final predictions or outputs are generated. By selecting the right output function, practitioners can ensure that the network's predictions are in the appropriate format for the given problem, leading to accurate and effective models.

2.6.3.1 Loss Functions

The neural network uses optimization strategies such as stochastic gradient descent to minimize the the algorithm's error. This error is calculated using a loss function. Loss functions are

crucial in training CNNs as they measure the difference between predicted values and ground truth. One commonly used loss function is the **Cross-Entropy Loss**, also known as the **Log Loss**. It is widely employed in classification tasks, both for binary and multi-class scenarios.

For binary classification, the binary cross-entropy loss is used. It is defined as follows:

$$\text{Binary Cross-Entropy Loss}(y, \hat{y}) = -\left(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right) \tag{2.32}$$

Where: - y represents the ground truth label (0 for one class and 1 for the other). - $\hat{y}$ represents the predicted probability of belonging to the positive class.

For multi-class classification, the categorical cross-entropy loss is commonly used. It is defined as:

$$\text{Categorical Cross-Entropy Loss}(y, \hat{y}) = -\sum_{i} y_i \log(\hat{y}_i) \tag{2.33}$$

Where: - y is a one-hot encoded vector representing the ground truth class. - $\hat{y}$ is a vector of predicted class probabilities.

These loss functions measure the dissimilarity between predicted and actual values and provide a gradient that guides the network during training. The goal is to minimize the loss, effectively improving the network's ability to make accurate predictions.

Function	***Formulas***	***Year***
Focal Loss	$FL\left(p_t\right) = -\left(1-p_t\right)^{\gamma}\log\left(p_t\right)$	2017
Cycle Consistency Loss	$l_{cyc}(G,F) = E_{x\sim Pdata(x)}\left[\left\|F\left(G(x)\right)-x\right\|_1\right] + E_{y\sim Pdata(y)}\left[\left\|G\left(F(y)\right)-y\right\|_1\right]$	2017
Triplet Loss	$L_t\left(v_p, v_n\right) = -\frac{1}{MN}\sum_i^M\sum_j^N logprob(vp_i, vn_j)$	2018
GAN Least Squares Loss	$\min\max V_{GAN}(D,G) = E_{x\sim Pdata(x)}\left[logD(x)\right]\left[\log(1*-D\left(G(z)\right))\right]$	2016
InfoNCE	$\mathcal{L}_N = -\mathbb{E}_X\left[\log\frac{f_k(x_{t+k}, c_t)}{\sum_{x_j\in X} f_k(x_j, c_t)}\right]$	2018
GAN Hinge Loss	$L_D = -\mathbb{E}_{(x,y)\sim p_{data}}[\min(0,-1+D(x,y))] - \mathbb{E}_{z\sim p_z, y\sim p_{data}}[\min(0,-1-D(G(z),y))]$ $L_G = -\mathbb{E}_{z\sim p_z, y\sim p_{data}} D(G(z),y)$	2017
GAN Hinge Loss	$L_D = -\mathbb{E}_{(x,y)\sim p_{data}}[\min(0,-1+D(x,y))] - \mathbb{E}_{z\sim p_z, y\sim p_{data}}[\min(0,-1-D(G(z),y))]$ $L_G = -\mathbb{E}_{z\sim p_z, y\sim p_{data}} D(G(z),y)$	2017
Dice Loss	$DiceLoss\left(y,\overline{p}\right) = 1 - \frac{\left(2y\overline{p}+1\right)}{\left(y+\overline{p}+1\right)}$	2017
ArcFace	$L_3 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))}+\sum_{j=1,j\neq y_i}^{n} e^{s\cos\theta_j}}$	2018
Supervised Contrastive Loss	$\mathcal{L}^{sup} = \sum_{i=1}^{2N}\mathcal{L}_i^{sup}$ **with** $\mathcal{L}_i^{sup} = \frac{-1}{2N_{\tilde{\boldsymbol{y}}_i}-1}\sum_{j=1}^{2N}\mathbf{1}_{i\neq j}\cdot\mathbf{1}_{\tilde{\boldsymbol{y}}_i=\tilde{\boldsymbol{y}}_j}\cdot\log\frac{\exp\left(\boldsymbol{z}_i\cdot\boldsymbol{z}_j/\tau\right)}{\sum_{k=1}^{2N}\mathbf{1}_{i\neq k}\cdot\exp\left(\boldsymbol{z}_i\cdot\boldsymbol{z}_k/\tau\right)}$	2020

Tableau 2.6: Top 10 of most used loss function

2.7 Overfitting and Underfitting in CNN

Overfitting happens when the neural network is good at learning its training set, but is not able to generalize its predictions to additional, unseen examples. This is characterized by low bias and high variance. Underfitting happens when the neural network is not able to accurately predict for the training set, not to mention for the validation set. This is characterized by high bias and high variance (details in Chapter1-Machine learning)

Methods to avoid Overfitting	Methods to avoid Underfitting
• **Retraining neural networks:**	• **Adding neuron layers or inputs:**
Running the same model on the same training set but with different initial weights, and selecting the network with the best performance.	Adding neuron layers, or increasing the number of inputs and neurons in each layer, can generate more complex predictions and improve the fit of the model.
• **Multiple neural networks:**	• **Adding more training samples or improving quality:**
Training several neural network models in parallel, with the same structure but different weights, and averaging their outputs.	The more training samples you feed into the network, and the better they represent the variance in the real population, the better the network will perform.
• **Early stopping:**	• **Dropout:**
Training the network, monitoring the error on the validation set after each iteration, and stopping training when the network starts to over-fit the data.	Randomly "kill" a certain percentage of neurons in every training iteration. This ensures some information learned is randomly removed, reducing the risk of overfitting.
• **Regularization:**	• **Decreasing regularization parameter:**
Adding a term to the error function equation, intended to decrease the weights and biases, smooth outputs and make the network less likely to over-fit.	Regularization can be overdone. By using a regularization performance parameter, you can learn the optimal degree of regularization, which can help the model better to fit in the data.
• **Tuning performance ratio:**	
Similar to regularization, but using a parameter that defines by how much the network should be regularized.	

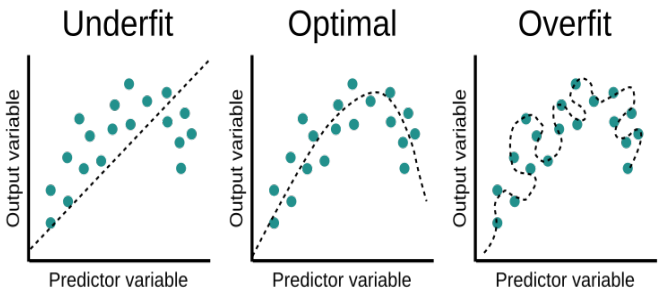


Figure 2.23: Over/Under/Optimal Fitting

2.8 Tips for CNN Architecture Design in Medical Imaging and COVID-19 Recognition

- **Data Augmentation**: Medical imaging datasets, especially for rare diseases like COVID-19, are often limited in size. Use data augmentation techniques such as rotation, flipping, scaling, and adding noise to artificially increase the diversity of your training data and improve the model's generalization ability.
- **Transfer Learning**: Transfer learning is particularly useful in medical imaging tasks where labeled datasets may be limited. Start with a pre-trained CNN model (e.g., ResNet, VGG, DenseNet) that was trained on a large dataset (e.g., ImageNet) and fine-tune it on your medical imaging dataset. This allows the model to leverage features learned from the pre-trained model and adapt them to your specific task.
- **Attention Mechanisms**: Consider incorporating attention mechanisms into your CNN architecture to highlight important regions of the input image that are relevant for making the diagnosis. Attention mechanisms can help the model focus on salient features while ignoring irrelevant or noisy information.
- **Ensemble Learning**: Ensemble learning involves combining predictions from multiple independently trained models to improve overall performance. Train multiple CNN architectures with different initializations, hyperparameters, or input representations, and combine their predictions through techniques like averaging or stacking.
- **Interpretability**: Interpretability is crucial in medical imaging tasks to gain insights into the model's decision-making process and build trust among clinicians. Use techniques such as gradient-based visualization methods (e.g., Grad-CAM), occlusion sensitivity, or saliency maps to visualize and interpret the model's predictions.
- **Class Imbalance Handling**: Medical imaging datasets often suffer from class imbalance, where certain classes (e.g., COVID-19 positive cases) are underrepresented. Use techniques like class weighting, oversampling, or data resampling to address class imbalance and prevent the model from being biased towards the majority class.
- **Multi-Modal Fusion**: In some cases, medical imaging datasets may contain information from multiple modalities (e.g., X-ray and CT scans). Design CNN architectures that can effectively fuse information from multiple modalities to improve diagnostic accuracy. This could involve using parallel branches for different modalities or incorporating multi-input architectures like Siamese networks.
- **Uncertainty Estimation**: Incorporate uncertainty estimation techniques into your CNN architecture to quantify the model's confidence in its predictions. This is particularly important in medical imaging tasks where incorrect diagnoses can have serious

consequences. Bayesian neural networks, Monte Carlo dropout, or ensemble methods can be used to estimate uncertainty.

- **Regularization Techniques**: Regularization techniques such as dropout, batch normalization, and weight decay are important for preventing overfitting, especially in scenarios with limited data. Experiment with different regularization techniques to find the optimal balance between model complexity and generalization performance.

- **Domain-Specific Preprocessing**: Preprocess medical imaging data with domain-specific techniques tailored to the characteristics of the imaging modality (e.g., CT, MRI) and the specific task (e.g., COVID-19 recognition). This may include techniques such as windowing, normalization, and artifact removal to enhance the quality of the input data.

2.9 Conclusion

In conclusion, convolutional neural networks (CNNs) have emerged as a powerful tool in the field of deep learning, revolutionizing various applications such as image recognition, object detection, and natural language processing. Throughout this chapter, we have delved into the intricate components and operations that constitute a CNN, providing a comprehensive understanding of its inner workings.

We began by exploring the fundamental building blocks of CNNs, including convolutional layers, pooling layers, and fully connected layers. These layers work synergistically to extract hierarchical features from input data, enabling the network to learn intricate patterns and representations.

Moreover, we discussed the importance of activation functions such as ReLU, which introduce non-linearity into the network, enhancing its expressive power and enabling it to model complex relationships within the data.

Furthermore, we examined the role of regularization techniques like dropout and batch normalization, which mitigate overfitting and improve the generalization capabilities of CNNs, ensuring robust performance on unseen data.

Additionally, we highlighted the significance of optimization algorithms such as Adam and its variants, which facilitate the training process by efficiently updating the network parameters based on the learning rate of the loss function.

After all these, we think now that we are ready to explore existing architectures or create our own model.

In summary, convolutional neural networks represent a cornerstone in the realm of deep learning, offering unprecedented capabilities in handling complex data modalities and achieving remarkable performance across a myriad of applications. As the field continues to evolve, it is imperative to stay abreast of the latest advancements and methodologies, leveraging the power of CNNs to tackle increasingly challenging tasks and propel the boundaries of artificial intelligence.

Chapter 3

The New approach based on light enhancement and real-time dual CNN for classification of COVID-19 X-ray images

3.1 Introduction

A novel coronavirus 2019 (COVID-19) outbreak emerged in Wuhan, China, in early 2020 and swiftly spread to more than 200 countries worldwide [87]. Particularly impactful in areas requiring assistance, the pandemic's exponential propagation reached millions daily. COVID-19, a devastating pandemic, continues to claim numerous lives daily. The global contamination has strained healthcare systems immensely. Presently, the virus affects individuals globally, with the absence of a dependable vaccine being a significant concern. COVID-19 transmission between individuals leads to the development of pneumonia within the body.

The considerable rise in the number of individuals infected, particularly those experiencing respiratory inflammations, coupled with the insufficient availability of medical personnel and healthcare facilities in certain regions, has exacerbated the strain on the healthcare system.

The necessity for novel, effective, and faster diagnostic approaches surpassing manual diagnosis conducted by experts has become imperative. Concurrently, the widespread adoption of rapid diagnostic tools, aiding in swift measurements and recommending appropriate treatments, underscores both the overwhelming impact of the pandemic and the effectiveness of these tools in curbing the virus's spread. Consequently, the conventional method of detecting new positive cases was not always feasible due to the time required when using PCR tests, as well as their limited availability and high costs. Similarly, CT scans, though effective, are prohibitively expensive, whereas X-rays, while readily accessible, lack the quality of CT images, necessitating the use of some extra processings.

Health professionals have identified respiratory tract disease and severe viral pneumonia with respiratory failure as clinical manifestations of COVID-19 infection [88, 89]. In a study

investigating 813 adult patients admitted to Wuhan Pulmonary Hospital [90], it was found that the majority of deceased patients were men. Nearly half of the cases presented with multiple disorders, with hypertension being the most prevalent, followed by coronary heart disease and diabetes, among others. Common symptoms upon hospital admission included cough and high fever, followed by sputum production and exhaustion [91]. In severe cases, the infection can lead to pneumonia, multi-organ failure, and ultimately, death.

Chest X-rays are commonly utilized in diagnosing COVID-19 infection, revealing characteristic white spots in the lungs of affected patients. While conventional diagnostic methods have improved in accuracy over time, they still pose risks to the safety of healthcare workers and are costlier due to the need for diagnostic test kits per patient. Conversely, medical imaging techniques such as X-rays and Computerized Tomography (CT) scans offer faster, safer, and more widely accessible screening options compared to traditional methods. X-rays are preferred over CT scans for COVID-19 screening due to their greater availability and lower cost [92, 93].

Integrating artificial intelligence (AI) into image processing holds promise for developing a highly efficient and precise technology capable of distinguishing between normal and diseased chest radiographs [94]. This approach has the potential to expedite the identification of COVID-19 cases and curb the spread of the disease, especially in resource-constrained settings lacking specialized radiologists and advanced medical equipment for early diagnosis and patient management.

The prospects of supervised machine learning (SML) in the healthcare sector, along with the challenges it encounters and potential solutions, are discussed in [95]. This paper highlights the opportunity for healthcare advancement through AI and SML in the near future, as these techniques have demonstrated effectiveness in diagnosing diseases with acceptable accuracy and high speed.

Recent advancements in computer vision (CV) and artificial intelligence (AI), particularly utilizing deep learning (DL) models [96], have shown promising results. Ensembles of convolutional neural networks (CNNs) have emerged as efficient tools for detecting skin cancer [97]. Additionally, numerous studies based on medical images address various applications such as pneumonia [98] and lung diseases [99].

Incorporating time series data, which often contain local and global patterns, has also been explored [100]. Hybrid approaches involving attention and LSTM networks have been proposed, with networks acting as relation extraction networks to uncover intrinsic relationships among features extracted from different data positions.

Convolutional neural networks (CNNs) have proven effective for medical image analysis [92], complementing the applications of computer vision and artificial intelligence in this domain [101].

In [102], a newly developed version of an algorithm is introduced to optimize the weights of a backpropagation neural network for the purpose of designing an effective segmentation tool. Additionally, [103] presents a robust image segmentation model tailored for cancer images, taking into account interval uncertainties. Furthermore, [104] proposes nerve optic segmentation

in CT images using a deep learning model in conjunction with a texture descriptor.

Machine learning techniques in the medical field extend beyond disease diagnosis to encompass medical image segmentation, as discussed in [105] and [106].

Symptoms of COVID-19 can vary among individuals, but commonly reported ones include fatigue, coughing, and shortness of breath. However, these symptoms overlap with those of other illnesses like pneumonia. Reverse transcription polymerase chain reaction (RT-PCR) tests are widely used and reliable for detecting the virus. Nevertheless, they have several limitations: they are slow, taking up to 24 hours for results, they put medical staff at risk due to physical contact, and they are costly, posing accessibility challenges in low-income countries. Therefore, there is a need for a quick, reliable, and affordable diagnostic method to prompt appropriate actions.

Chest radiography (chest X-ray) is another frequently used method for diagnosing lung diseases and detecting COVID-19. However, it also has drawbacks. Interpretation of X-ray images requires expertise, and false results can occur due to similarities between the chest X-ray images of COVID-19 patients and those with different types of pneumonia.

Numerous research efforts have been dedicated to diagnosing COVID-19 diseases, yet many of these studies assume readily available, clean, and accurately labeled data. However, a significant drawback is the lack of generalization to external data or real-world cases. Moreover, the discussion on the number of parameters in the proposed models is often absent, raising uncertainties about their performance in resource-constrained environments. Therefore, this study aims to address the following research questions:

Question 1: What is the optimal amount of data required to train a lightweight model relative to its complexity?

Question 2: Can automatic segmentation of the lungs and region labeling enhance feature extraction for improved classification?

Question 3: Does preprocessing steps, such as grayscale enhancement, contribute to increased accuracy in the classification task?

Question 4: How does the performance of the proposed CNN model compare to benchmark models in detecting COVID-19 lung disease in classification tasks?

Question 5: Can a lightweight model with a reduced number of parameters and low computational cost achieve satisfactory performance for this classification task?

To address the aforementioned challenges, this thesis proposes a novel approach for analyzing COVID-19 chest X-rays utilizing a deep learning model. The process begins with a preprocessing step aimed at enhancing image quality through the application of the color balance algorithm. This step ensures improved readability of tissues without altering the original data. Subsequently, lung segmentation is conducted using the UNet architecture to eliminate irrelevant data and enhance learning. To mitigate the effects of class imbalance, data augmentation techniques are applied.

The core of the system is a lightweight CNN model designed to achieve high generalization performance even with a small training dataset. This characteristic of being lightweight

holds significant practical implications, as it enables the proposed model to be deployed on devices with limited processing capabilities, such as edge devices and single-board computers, facilitating real-time operation.

The classification performance of the proposed architecture was evaluated and compared with widely-used CNN models, including DenseNet, ResNet, InceptionV3, VGG16, and VGG19. These benchmark models vary in design, number of parameters, and depth, enabling a comprehensive comparison with the proposed model. By examining their performance metrics, such as accuracy, precision, recall, and F1-score, the effectiveness of the proposed architecture can be assessed relative to these established models.

After The first Section of the Introduction, the remainder of this chapter is structured as follows:

Section 2: Related Work - This section provides an overview of previous research and approaches relevant to the proposed study, and attack their limitations.

Section 3: Proposed Approach - Here, the methodology and approach employed in the study are detailed, including the preprocessing steps, model architecture, and evaluation metrics.

Section 4: Experiments and Results - This section outlines the experimental setup, including dataset used, training procedure, and model evaluation. Additionally, the results of the experiments are presented and discussed.

Section 5: Conclusion - The paper concludes with a summary of the key findings, contributions, and potential future directions for research in this area.

3.2 Related work

Despite vaccination against its most recent mutation, COVID-19 is still wreaking havoc around the world. To avoid the spread of the virus and avoid complications, good pandemic management necessitates early diagnosis. A viral test, which is among the best diagnostic tools, can provide early detection. In addition to the lack of availability, the fastest viral test (RDT) takes about a half-hour to produce results and cannot distinguish between viral infections. The commonly used RT-PCR produces fewer false-positive results than RDT, but it has low sensitivity and requires more than 14 days to ensure the test's accuracy. Thus, given the problems with the viral test, chest X-rays may be the best low-cost alternative for detecting COVID-19. Furthermore, most hospitals have the necessary image acquisition equipment. The availability of such easily accessible data, combined with the power of deep learning-based image processing models, Due to the automated feature extraction capability of convolutional neural networks (CNN), compelled the researchers to pursue this avenue. A review of relevant papers revealed that transfer learning with well-known CNN architectures trained on ImageNet was used in more than 75% of the works, considering a binary problem to detect healthy and COVID-19 X-ray images.

ResNet has been extensively employed for detecting COVID-19 in chest X-ray images. In [107], the authors proposed enhancing COVID-19 chest X-ray diagnostics by employing drop-

weights based Bayesian Convolutional Neural Networks (BCNN). Additionally, [108] utilized SVM in conjunction with ResNet50 to classify X-ray images affected by coronavirus.

In [109], the authors introduced a solution for automated screening of coronavirus disease based on chest X-rays by combining the advantages of fine-grained classification and cost-sensitive learning. They demonstrated that the proposed discriminative cost-sensitive learning (DCSL) approach can be applied to any deep neural network.

Another ResNet-based approach was presented by [89], where three pre-trained convolutional neural network-based models (ResNet50, ResNet101, and ResNet152) were investigated for detecting pneumonia-infected patients using chest X-rays.

In [110], the authors developed a new deep anomaly detection model for fast and reliable screening. They proposed a one-class classification-based anomaly detection approach, focusing on distinguishing viral pneumonia cases from non-viral pneumonia and healthy controls. The proposed method utilized confidence-aware anomaly detection to determine the presence of viral pneumonia.

In [111], the authors utilized a Convolutional Neural Network (CNN) with the ResNet-101 model for COVID-19 diagnosis. They employed a dataset comprising 2562 images categorized as positive COVID-19 (1281 images) and negative COVID-19 (1281 images). A preprocessing step was applied to the dataset using Contrast Limited Adaptive Histogram Equalization (CLAHE). The results demonstrated that this preprocessing technique improved the model's performance, with an accuracy of 99.61% compared to 99.22% accuracy achieved with the raw data.

Additionally, [112] proposed an integrated method for selecting the optimal deep learning model for COVID-19 diagnosis based on a novel crow swarm optimization algorithm. Among the models evaluated, the ResNet50 model achieved the highest accuracy of 91.46%.

In [113], the authors employed a pre-trained Convolutional Neural Network (CNN) with COVID-19 and pneumonia cases.

In [114], several training techniques were introduced to enhance the learning process of deep convolutional networks when the dataset contains a limited number of samples. Furthermore, this work proposed a hybrid system that combines Xception and ResNet50V2 networks for feature extraction.

Moreover, [115] tackled the issues of memory and processing time by utilizing an extended version of EfficientNet, which was identified as the most efficient in terms of accuracy and low resource usage.

A new deep domain adaptation method was introduced in [116], this approach aimed to adapt pneumonia knowledge for COVID-19 diagnosis through a novel classifier separation model. This model utilized the differences between domains to adapt contradictory functionalities.

Furthermore, [117] explored the capabilities of deep learning on chest radiographs by proposing a cascade approach using SEME-ResNet50 and SEME-DenseNet169. This work also utilized MoEx and histogram equalization for data enhancement. Additionally, the extracted features of the model were enhanced by employing the Regular Sparse Model [118].

In [119], the authors addressed the challenge of a small number of trainable data by proposing a patch-based convolutional neural network approach inspired by the statistical analysis of potential CXR imaging biomarkers.

Similarly, in [120], the authors tackled the same problem and compared various state-of-the-art deep learning approaches with a random oversampling and weighted class loss function approach for unbiased fine-tuned learning.

Given the difficulty of collecting medical-related training data, [121] proposed the use of federated learning for COVID-19 data training. This approach allows for the development of a shared model without accessing local data.

In [122], the authors introduced a multi-label classification approach using five deep learning models, namely ResNet18, ResNet34, InceptionV3, InceptionResNetV2, and DenseNet161, to predict multiple pathologies for each patient. Furthermore, pathology prediction can benefit from the application of the latest metaheuristic methods [123–128].

To enhance detection accuracy, [14] proposed building a new chest X-ray dataset from various public databases and developing a COVID-19 detection system based on transfer learning with pre-trained deep learning algorithms.

In [129], the authors introduced a novel 3-step fine-tune technique to enhance performance and reduce training time on a pre-trained ResNet-50 architecture.

[130] employed Generative Adversarial Networks (GAN) with fine-tuned deep transfer learning to detect pneumonia in chest X-ray images for a limited dataset.

To extract the most accurate features, a selection of popular deep convolutional neural networks was utilized in [91]. These features were then employed in other machine learning tools for classification purposes.

A new deep learning framework named COVIDX-Net was proposed in [131,132]. COVIDX-Net utilized seven different CNNs to better analyze the intensities of X-ray images and classify patients accordingly.

In [133], the authors addressed the categorization of COVID-19 severity into mild, moderate, and severe. They utilized a deep learning model for pneumonia classification.

Transfer learning was utilized on a subset of 2000 chest X-ray images in [134], where four popular convolutional neural networks were trained. Moreover, the authors focused on dataset improvement, introducing a new dataset consisting of 5000 chest X-ray images.

In [135], a compact classifier (CSEN) was proposed for early-stage detection of COVID-19 based on chest X-ray images.

Lastly, [108] proposed a deep learning-based methodology coupled with an SVM classifier to detect affected X-ray images.

DenseNet has played a significant role in the development of COVID-19 detection systems on chest X-ray images.

In [136], a total of 3000 chest X-ray images, including 1000 healthy cases, were prepared. The authors proposed fine-tuning three pre-trained CNN models to detect abnormalities in these images.

A novel approach called GSA-DenseNet121-COVID-19 was introduced in [137]. This involved using a CNN model in conjunction with an optimization algorithm called GSA to assist the classifier in determining the best values of hyperparameters.

To address the challenge of a small number of samples in public data, cascaded learning strategies were proposed in [138–140]. These approaches aimed to improve both the sensitivity and specificity of the classifier. Additionally, the proposed models were trained on a large dataset of chest X-rays depicting non-COVID-19 pneumonia.

In [140], a fine-tuned resident fellow (RF) network was utilized to discriminate COVID-19 cases from normal cases, despite the small number of COVID-19 cases available for training.

Furthermore, four VGG-based approaches were investigated in [141–143].

In [143], the focus was on addressing the challenge of a small number of training data, while [141] proposed a novel approach consisting of three phases: pneumonia detection, COVID-19 recognition, and disease localization.

According to the results of [144], VGG16 and VGG19 achieved the highest accuracy score of 95

In the paper "Detecting Covid19 and pneumonia from chest X-ray images using deep convolutional neural networks," Transfer Learning models like VGG16 and ResNet50 were utilized, achieving an accuracy of 89.34

To improve accuracy, several image pre-processing algorithms were employed to remove diaphragms and normalize image contrast and ratio [142].

Other deep learning models with modest usage in the state of the art include Inception and Xception [145–147], and AlexNet [148, 149].

It's true that transfer learning, while commonly used in deep learning, may not always be the most suitable approach for specific applications like COVID-19 detection in chest X-ray images. Models pre-trained on large datasets with diverse object categories might not effectively capture the unique features relevant to COVID-19 diagnosis. Moreover, the size and quality of training datasets are crucial for the success of deep learning models. However, acquiring large and diverse datasets, especially in medical imaging, is challenging due to various obstacles, including privacy concerns regarding patient data. Therefore, while transfer learning can offer advantages such as faster training times and leveraging pre-existing knowledge, it's essential to carefully consider the suitability of pre-trained models for the specific task at hand. In domains like medical imaging where nuances and context are critical, fine-tuning or training models from scratch using domain-specific data may be more appropriate. Additionally, addressing challenges related to data privacy and acquisition is crucial for advancing research in this field.

Combining multiple models for improved results in COVID-19 detection is a common approach, with success heavily reliant on decision methods and the confidence level of each classifier.

In [150], the authors proposed various methods that integrate image processing and classifiers such as K-Nearest Neighbor (KNN) and Support Vector Machine (SVM) for classification and early detection of COVID-19. The approach that combines Local Binary Pattern (LBP)

and KNN achieved the highest accuracy of 99%.

In [132], the authors employed several deep convolutional neural network models including InceptionV3, ResNetV2, InceptionResNetV2, DenseNet201, VGG19, MobileNetV2, and Xception to classify X-ray images as positive or negative for COVID-19. The highest classification performance, with an accuracy of 90%, was achieved using VGG19 and DenseNet201.

[151] used different pre-trained CNN models: AlexNet, VGG-16, MobileNet-V2, SqueezeNet, ResNet-34, ResNet-50, and COVIDX-Net. The dataset contains 406 images evenly distributed between COVID-19 and healthy classes. ResNet-34 achieved the best prediction accuracy of 98.33%.

[152] proposed a new CNN model to detect COVID-19. The dataset contains 4316 chest X-ray images (2158 COVID-19 negative scans and 2158 COVID-19 positive scans), and a data augmentation technique was used. The model achieved an accuracy of 95.5%.

In addition to the use of fine-tuned available transfer learning models, there are some other studies in which specific CNN architectures are proposed.

In [153], the authors proposed a new CNN model for detecting COVID-19 and influenza cases. The model achieved an accuracy score of 93% on a dataset of 8304 X-ray images.

The authors of [154] proposed an approach comprised of several stages. They considered a feature fusion-based EfficientNetV2 and then ResNet101, along with Convolutional Block Attention Module and SVM classifiers, respectively. Data augmentation was applied, and the results showed that the system achieved an accuracy of 99.89%.

In [155], the authors implemented a pre-trained InceptionV3 scheme with chosen multi-class classifiers to detect the disease and assess its severity level. The dataset contains four classes (Normal, Mild, Moderate, and Severe Pneumonia). The best result achieved in this work reached an accuracy of 85.18%.

A CNN architecture proposed by [156] was tested on a dataset containing 15,475 chest X-ray images and on another (Enhanced COVID-19) including 1092 chest X-ray images (364 images for each class). Data augmentation was applied to datasets, and the class weight method was applied to rebalance the datasets. The results showed that the proposed model achieved an accuracy of 94% for the dataset and 99% for the enhanced datasets.

[157] proposed in 2023 a COVID-19 diagnosis from chest X-ray images using CNNs; they achieved an accuracy value of 89.89%.

Typically, deeper models with more convolutional layers may achieve better feature extraction, eventually yielding more successful classification. However, such models have major drawbacks like the requirement of large amounts of images and expensive hardware with heavy computational capability. Thus, in situations like the pandemic, with a low number of verified samples, such models may lead to lower generalization, which is a significant problem.

we present the taxonomy of COVID-19 detection studies based on the method/model used, dataset size, and accuracy percentage reported in each study, in Table 3.1.

Tableau 3.1: Taxonomy of related Work in COVID-19 Detection Using Deep Learning Models

Study Reference	Used Model	Year	Metrics	Approach	various
[107]	ResNet	2020	Accuracy: 99%	Drop-weights based Bayesian CNN (BCNN)	N/A
[108]	SVM + ResNet50	2020	Accuracy: 91.39%	Fine-grained classification, Cost-sensitive learning	N/A
[109]	Fine-grained classification + Deep learning	2020	Accuracy: N/A	Discriminative cost-sensitive learning (DCSL)	N/A
[89]	ResNet50, ResNet101, ResNet152	2020	Accuracy: N/A	Deep learning-based models	N/A
[110]	Deep anomaly detection model	2020	Accuracy: N/A	One-class classification, Anomaly detection	Confidence-aware anomaly detection
[111]	ResNet-101	2020	Accuracy: 99.61%	Preprocessing technique (CLAHE)	N/A
[112]	ResNet50	2020	Accuracy: 91.46%	Crow swarm optimization algorithm	N/A
[113]	Pre-trained CNN	2020	Accuracy: N/A	Pre-trained CNNs	N/A
[114]	Xception + ResNet50V2	2020	Accuracy: N/A	Hybrid system, Feature extraction	N/A
[115]	Extended EfficientNet	2020	Accuracy: N/A	Extended EfficientNet	Memory and processing time
[116]	Convolutional Neural Network (CNN)	2020	Accuracy: N/A	Deep domain adaptation method	N/A
[117]	SEME-ResNet50, SEME-DenseNet169	2020	Accuracy: N/A	Cascade approach, MoEx, Histogram equalization	Regular Sparse Model

Tableau 3.1: Taxonomy of related Work in COVID-19 Detection Using Deep Learning Models

Study Reference	Used Model	Year	Metrics	Approach	various
[119]	Patch-based CNN	2020	Accuracy: N/A	Patch-based approach	Small number of trainable data
[120]	Various deep learning approaches	2020	Accuracy: N/A	Random oversampling, Weighted class loss	Small number of trainable data
[121]	Federated learning	2020	Accuracy: N/A	Federated learning	Difficulty of collecting medical-related training data
[122]	ResNet18, ResNet34, InceptionV3, InceptionResNetV2, DenseNet161	2020	Accuracy: 95%	Multi-label classification approach	Application of metaheuristic methods
[14]	Transfer learning models	2020	Accuracy: N/A	New chest X-ray dataset, Transfer learning	N/A
[129]	Fine-tuned ResNet-50	2020	Accuracy: N/A	Fine-tune technique	N/A
[130]	Generative Adversarial Networks (GAN)	2022	Accuracy: N/A	Fine-tuned deep transfer learning	N/A
[91]	Popular deep CNNs for feature extraction	N/A	N/A	Feature extraction	N/A
[131], [132]	COVIDX-Net	N/A	N/A	Seven different CNNs for classification	N/A
[133]	Deep learning model	N/A	Accuracy: 95.5%	Severity classification, Pneumonia classification	N/A
[134]	Transfer learning models	N/A	Accuracy: N/A	Data augmentation, Dataset improvement	Small number of samples in public data

Tableau 3.1: Taxonomy of related Work in COVID-19 Detection Using Deep Learning Models

Study Reference	Used Model	Year	Metrics	Approach	various
[135]	Compact classifier (CSEN)	N/A	Accuracy: N/A	Early-stage detection, Chest X-ray images	N/A
[141], [142], [143]	VGG-based approaches	N/A	Accuracy: N/A	Various approaches	Small number of training data
[144]	VGG16, VGG19	N/A	Accuracy: 95%	Evaluation of VGG models	N/A
[150]	Image processing + KNN, SVM	N/A	Accuracy: 99%	Integrated methods	N/A
[151]	Pre-trained CNN models	N/A	Accuracy: 98.33%	Various pre-trained CNN models	N/A
[152]	New CNN model	N/A	Accuracy: 95.5%	Data augmentation, New CNN model	N/A
[153]	New CNN model	N/A	Accuracy: 93%	New CNN model	N/A
[154]	EfficientNetV2, ResNet101	2021	Accuracy: 99.89%	Feature fusion-based EfficientNetV2	Data augmentation
[155]	Pre-trained InceptionV3	2022	Accuracy: 85.18%	Multi-class classifiers, Disease severity	N/A
[156]	Proposed CNN architecture	N/A	Accuracy: 94%	Proposed CNN architecture	Data augmentation, Class weight method
[157]	CNNs	2023	Accuracy: 89.89%	COVID-19 diagnosis using CNNs	N/A

As a consequence, robust and accurate detection of COVID-19 from chest X-ray images remains a challenge due to three major issues:

(a) COVID-19 exhibits similar symptoms to other types of pneumonia on chest X-rays,

(b) Misdiagnosis of COVID-19 is common, and the associated costs are high,

(c) And, most of the works that proposed new models fail to consider the computational load they impose.

Although some studies have made significant progress, they do not adequately address the three issues raised.

Therefore, these situations have been addressed to some extent in this study by proposing a model with reduced convolutional layers as well as the number of weights. Hence, it becomes more suitable for the detection task to be executed on a larger scale of digital devices including those with relatively lower computational power.

3.3 Proposed approach

This section discusses the workflow used to identify radiological images due to coronavirus. Figure 3.1 illustrates the overall approach.

Main Steps of our System:

(a) **Data Collection:** Gather and annotate a small dataset of images relevant to our task. Annotating involves labeling objects or regions of interest for segmentation.

(b) **Data Preprocessing and Augmentation:** Enhance the dataset by applying basic preprocessing techniques like resizing, normalization, and data augmentation (e.g., rotation, flipping, ...) to increase diversity in our limited dataset.

(c) **Model Selection:** Choose a lightweight Convolutional Neural Network (CNN) architecture suitable for our task. Some popular options include VGG16, VGG19, DenseNet, ResNet or InceptionV3.

(d) **Model Architecture:** Design the architecture for our UNet segmentation model, which typically consists of an encoder-decoder structure with skip connections and the lightweight Convolutional Neural Network (CNN), which typically consists of a stack of Convolution layers and other layers, that will be detailled in this chapter.

(e) **Model Training:** Train the UNet model using the segmented lungs dataset. And employ transfer learning by using a pretrained lightweight CNN to improve model performance.

(f) **Loss Function:** Define an appropriate loss function for our segmentation/classification task.

(g) **Training Strategy:** Train the model with a small learning rate, early stopping, and monitor its performance using metrics like Intersection over Union (IoU) for the segmentation or Precision, Accuracy, F1 score and other metrics for the detection and classification model.

(h) **Evaluation:** Evaluate our model on a separate validation set to measure its performance in terms of accuracy, precision, recall, and other relevant metrics.

(i) **Deployment:** Once we are satisfied with the model's performance, deploy it for inference on new images.

(j) **Fine-tuning:** Continuously improve the model's performance by fine-tuning it with any new data we collect or by adjusting hyperparameters.

(k) **Monitoring:** Monitor the model's performance and retraining or adapting to the task and dataset.

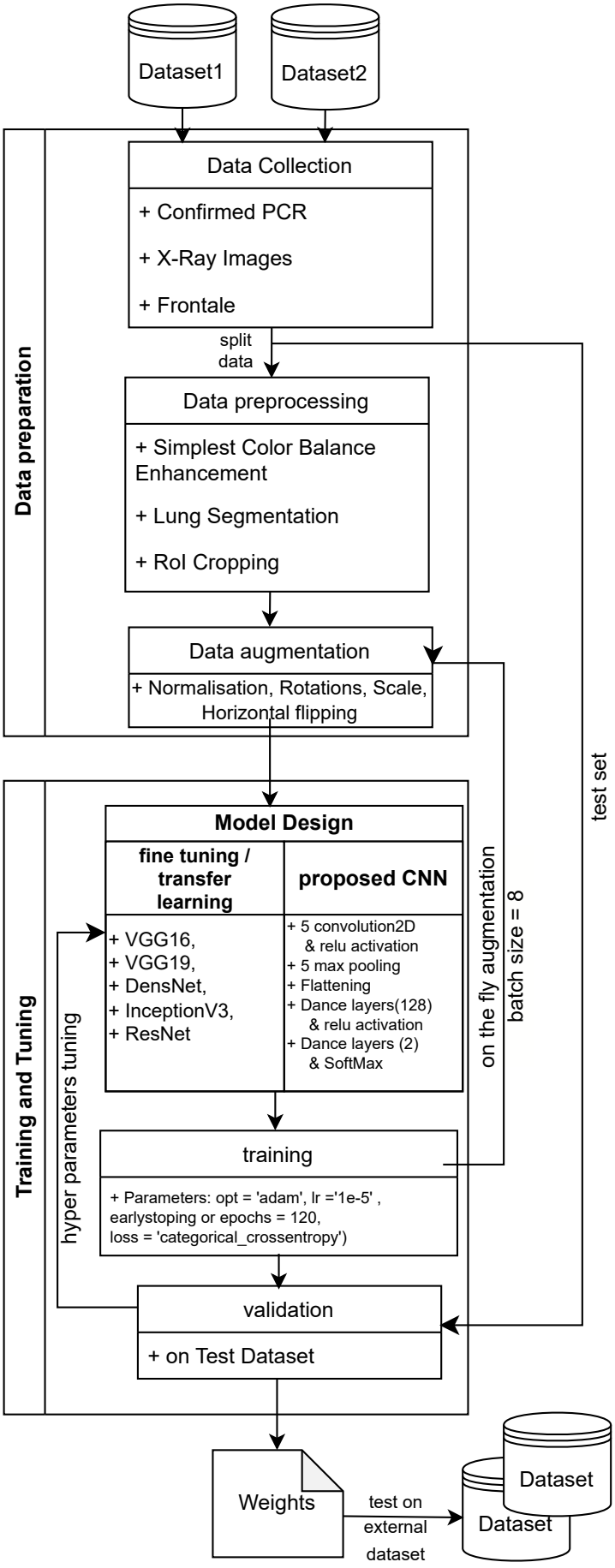


Figure 3.1: Workflow of the proposed approach

3.4 Data preparation

Choosing the dataset wasn't a simple task because we wanted to create a database with a large number of images to detect illness in Xray images.

However, we were unable to do so because in the first idea, We thought about using real patient cases from algerian hospitals, but that was near impossible.

Then in the second idea, we tried to find some relevant benchmark datasets, and caused by the urgent time we coulden't wait to collect, Unfortunately, we encountered a problem where some of the datasets didn't have annotations, while others didn't grant us permission to access them.

that's why we thought to combine some little confirmed dataset and to use the few images we found and to work on, in every step from the processing to the ROI selection to the augmentation and to the chosen model to train the dataset on it.

to validate the results, we tried to access more data from hospitals but due to privacy and confidentiality issue. we didn't had much data to train on,that's why we opted for the open access data from internet, and inspected every single image and kept only PCR verified x-ray images, and at the end we generalized on five external datasets from unseen images and our model gave the best True positive values and the less miss-classification rate.

3.4.0.1 exemple of covid19 case

Four years following the coronavirus pandemic, researchers continue to uncover more about the virus's enduring effects, including a phenomenon often termed "COVID lung." If you're still under the impression that COVID-19 is merely similar to the flu, consider this perspective. Dr. Brittany Bankhead-Kendall, a trauma surgeon from Texas, highlights a notable observation: it's exceptionally rare to encounter X-rays of COVID-19 patients without evident dense scarring.

She said, "Post-COVID lungs look worse than any type of terrible smoker's lung we've ever seen. And they collapse. And they clot off. And the shortness of breath lingers on... & on... & on."

An x-ray of healthy lungs shows mostly black, which is air. In a smoker's lungs, white lines are signs of scarring and congestion. Meanwhile, an x-ray of Covid lungs looks almost completely white 3.2.

Normal Lung Covid Lung Smoker's Lung

Figure 3.2: Example of Normal/Covid/Smoker's Lung

- **Healthy Lungs (Left):** The healthy lung on the left exhibits a uniform appearance with slight gray shading, indicating ongoing gas exchange processes.
- **COVID-19 Affected Lungs (Middle):** In contrast, the lungs of a COVID-19 patient in the middle appear more opaque and white, indicating inflammation and an ongoing infection.
- **Smoker's Lungs (Right):** On the right, the lungs of a smoker may display areas of dark black, suggesting the presence of emphysema.

and this is an example (Figure 3.3) of a patient's case evolution from day one to last day:

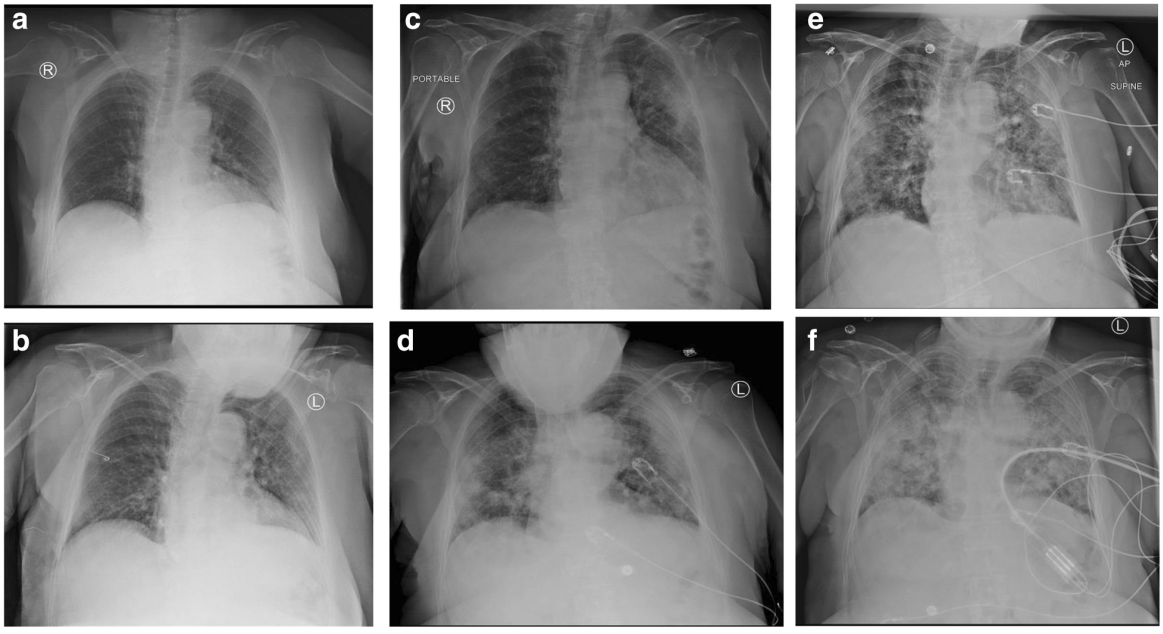


Figure 3.3: X-ray images taken sequentially for an 80-year-old patient diagnosed with COVID-19.

(a) Chest X-ray obtained on illness day 5 showed peripheral abnormalities.

(b) Chest X-ray obtained on illness day 7 showed an increased extent diffusely involving the left lung.

(c) Chest X-ray obtained on illness day 11 showed an increased extent involving the right lung, with increased consolidation diffusely involving the left lung.

(d) Chest X-ray obtained on illness day 14 showed the development of reticulations in both lungs with an increased extent of illness.

(e) Chest X-ray obtained on illness day 17 showed extensive bilateral consolidations mainly peripherally with increased reticulation.

(f) Chest X-ray obtained on illness day 18 showed extensive consolidation diffusely involving both lungs. The patient died on illness day 18.

3.4.1 challenges faced in data preparation:

Data preparation and processing for COVID-19 datasets can pose several challenges due to various factors such as the nature of the data, labling, its sources, and the rapidly evolving situation. Some common challenges faced in data preparation for COVID-19 datasets include:

- **Labeling and Annotation:** Accurate and reliable labeling of X-ray images is crucial for training machine learning models. Obtaining annotated datasets, especially for COVID-19 cases, can be challenging due to the need for expert radiologists and the potential scarcity of labeled data.
- **Images per class:** It ensures that the model receives sufficient training examples for each class, allowing it to learn and recognize objects accurately across all categories. $\geq$ 1500 images per class recommended and if there is less, data augmentation is a good solution.
- **Instances per class:** It ensures that the model receives sufficient examples of each class during training. The number of instances (labeled) per class recommended depends on the complexity of the object.
- **Image variety:** To ensure the dataset accurately reflects the environment where the model will be deployed, it is important to include images that capture various factors such as different period of illness, lighting conditions, angles, scale, and sources.
- **Class Imbalance:** Imbalances in the distribution of COVID-19 positive and negative cases can affect the performance of machine learning models. There might be a scarcity of positive cases, making it challenging for the model to learn and generalize effectively.
- **Data Augmentation:** Augmenting the dataset to increase its size and diversity is important for improving model robustness. However, applying traditional data augmentation techniques to medical images requires careful consideration to avoid introducing unrealistic variations.

- **Quality of Images:** Variability in image quality and resolution across different medical imaging devices and settings can impact the model's performance. Ensuring consistency in image quality and preprocessing steps is crucial for reliable analysis.
- **Data Cleaning and Preprocessing:** Outliers, duplicates, and inconsistencies must be identified and addressed to ensure the quality of the data. This may involve complex cleaning and preprocessing techniques to handle noisy or anomalous data points.
- **Ethical Considerations:** Ensuring patient privacy and complying with ethical standards is paramount. Anonymizing patient information and handling sensitive medical data in accordance with regulations are critical aspects of data preparation.
- **Transfer Learning Challenges:** Applying transfer learning from pre-trained models on non-medical datasets may not be straightforward due to the distinct characteristics of medical images. Fine-tuning models for medical imaging applications requires domain-specific expertise.
- **Temporal Aspects:** X-ray images taken at different time points during a patient's illness may reveal evolving patterns. Handling temporal aspects and changes in the appearance of lung abnormalities over time is a consideration in data preparation.
- **Validation and Generalization:** Evaluating the performance of models on diverse datasets, including data from different hospitals or regions, is important for generalizability. However, variations in imaging protocols and patient populations can complicate this process.
- **Integration with Clinical Data:** Combining X-ray image data with clinical information, such as patient history and laboratory results, can enhance the overall understanding of COVID-19. However, integrating heterogeneous data types requires careful consideration of data compatibility and preprocessing.

Addressing these challenges involves collaboration between data scientists, radiologists, and healthcare professionals to ensure that models are reliable, interpretable, and can contribute meaningfully to the diagnosis and management of COVID-19 cases. Additionally, following ethical guidelines and regulations for handling medical data is crucial throughout the entire data preparation process.

3.4.2 Data Collection

Deep learning based approaches need good datasets to ensure the effectiveness and credibility of the model. We conducted an extensive search on the Internet to find datasets that meet training requirements.

The target of this thesis is the categorizing Covid-19 disease on the basis of chest radiography. The images must be thoracic and frontal, and most importantly, they must be confirmed

by PCR test. Since searching under these circumstances is rather difficult, we used data from two datasets.

Indeed, having very few examples to learn from for a classification problem is a difficult machine learning problem, but it is also a realistic one: in many real-world use cases, even small-scale data collection can be expensive, if not impossible (e.g., in medical imaging). Our approach to data collection and processing is based on the ability to make the most of limited data.

Most deep learning neural network models rely on the amount and diversity of training data, and they are only accurate if the data is varied and "sufficient" to teach the model classes. When training a deep neural network algorithm and overcoming the challenge of under-fitting, the best advice is to have between 1000 and 5000 images per class, but what if we don't have much training data to begin with? We cannot expect the CNN to generalize after being trained on a small amount of data (Table 3.2).

this thesis aims to categorize COVID-19 disease based on chest radiography. The images must be ***thoracic*** and ***frontal***, and most importantly, they must be ***confirmed by PCR***. Since searching under these circumstances is rather difficult, we used data from two data sets:

- **Dataset 1 :** We took the class for *COVID-19* in the covid-chestxray-dataset [19] for the reason that it is rich in labeled images with all its features.

Type	Genus or Species	Image Count
Viral	COVID-19 (SARSr-CoV-2)	468
	SARS (SARSr-CoV-1)	16
	MERS-CoV	10
	Varicella	5
	Influenza	4
	Herpes	3
Bacterial	*Streptococcus* spp.	13
	Klebsiella spp.	9
	Escherichia coli	4
	Nocardia spp.	4
	Mycoplasma spp.	5
	Legionella spp.	7
	Unknown	2
	Chlamydophila spp.	1
	Staphylococcus spp.	1
Fungal	*Pneumocystis* spp.	24
	Aspergillosis spp.	2
Lipoid	Not applicable	8
Aspiration	Not applicable	1
Unknown	Unknown	59

Figure 3.4: types of pulmonary disease in covid-chestxray-dataset.

this dataset is open source and contain chest X-ray and CT images of patients which are positive or suspected of COVID-19 or other viral and bacterial pneumonias (MERS, SARS,

Tableau 3.2: Most of the used COVID-19 datasets

#	Dataset	Description	Total	Normal	Covid	Pneumonia
1	COVID-19 Radiography Database [14]	Chest X-ray database COVID-19	21200	10192	1616	1345
2	COVID-19 Chest X-ray [15]	Collection of image data COVID-19	357	0	357	0
3	COVID-19 Image Dataset [16]	3-way classification - COVID-19, viral pneumonia, normal	317	90	137	90
4	COVIDx CXR-2 [158] [159]	Chest X-rays for COVID-19 detection	16400	0	16400	0
5	COVID-19 X-ray Dataset (Train & Test Sets) [160]	Pneumonia detector COVID-19 CNN	188	94	0	94
6	COVID-19 Patients Lungs X Ray Images 10000 [161]	Dataset on the Corona virus	98	28	70	0
7	Chest X-ray (COVID-19 & Pneumonia) [162]	The data set contains chest radiographic images of COVID-19, pneumonia, and normal patients	6432	1583	576	4273
8	COVID-19 Chest X-ray Image Dataset [163]	normal and COVID-19 affected chest x-ray images	94	25	69	0
9	Chest X-ray for COVID-19 detection [17]	Chest X-ray data set for the detection of COVID-19	371 (23val)	174	174	0
10	CoronaHack -Chest X-Ray-Dataset [164]	Classer l'image X Ray qui a Corona	5935	0	5935	0
11	COVID19 Pneumonia Normal Chest X-ray PA Dataset [165]	COVID19 data set with pneumonia and normal chest radiograph (PA)	6939	2313	2313	2313
12	COVID-19 X rays [166]	X-rays and CT scans of COVID-19 patients	97	0	97	0
13	Covid Patients Chest X-Ray [18]	Download 162 images of chest x-rays of covid and normal patients	342	162	162	0
14	Covid-GAN and Covid-Net mini Chest X-ray [167]	Chest X-ray for patients with covid and pneumonia with generic GAN	7544	2083	972	4489
15	Chest X-Ray Images (Pneumonia) [168]	5,863 images, 2 catégories	5856	1583	0	4273
16	covid-chestxray-dataset [19]	Chest X-ray and CT image data set of patients positive or suspected of having COVID-19 or other viral and bacterial pneumonias)	930	0	930	0

and ARDS) as we can see in table 3.4 [19]. we have chosen only the COVID-19 cases, and some examples are in figure 3.3.

we noticed that in the metadata file metadata link, it contains hundreds of images from every continent and many countries to give a rich and divers samples of divers ages, sexes, backgrounds of confirmed ill/infected persons.

To validate the results, we tried to access more data from hospitals but due to privacy and confidentiality issue. we didn't had much data to train on, that's why we inspected every single image and kept only PCR verified and frontal x-ray images.

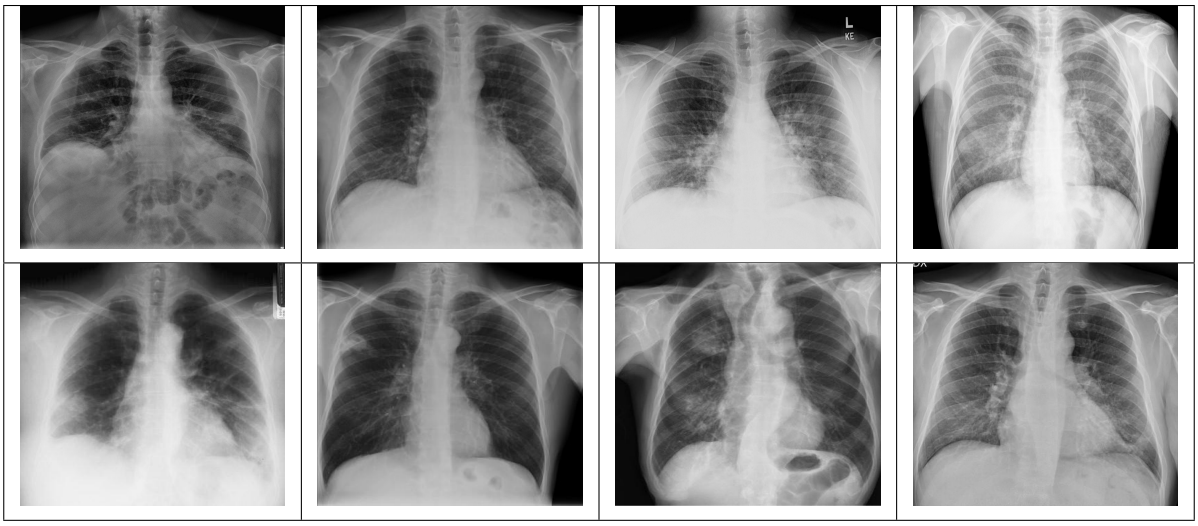


Tableau 3.3: Some samples of COVID-19 from [19] dataset.

- **Dataset 2 :** As for the data for *normal* people, we took the class from chestxray dataset [168].

The dataset is organized into 3 folders (train, test, val) and contains subfolders for each image category (Pneumonia/Normal). There are 5,863 X-Ray images (*.JPEG) and 2 categories (Pneumonia/Normal).

Normal category of Anterior-posterior chest X-ray images were chosen from retrospective patient cohorts at Guangzhou Women and Children's Medical Center. The imaging was conducted as a routine part of the patients' clinical care.

To ensure the reliability of the chest X-ray analysis, all radiographs underwent an initial quality control screening, eliminating any scans of low quality or unreadable nature. Subsequently, two expert physicians graded the diagnoses of the images before they were deemed suitable for training the AI system. As a precaution against grading errors, a third expert also reviewed the evaluation set.

3.4.3 Data Cleaning

As a result, a new reliable and confirmed dataset was created. It contains two classes: one named "Covid" for the radiological chest images of Covid patients and the other named "Normal"

for the radiological chest images of normal patients. Both classes contain 200 unaugmented images.

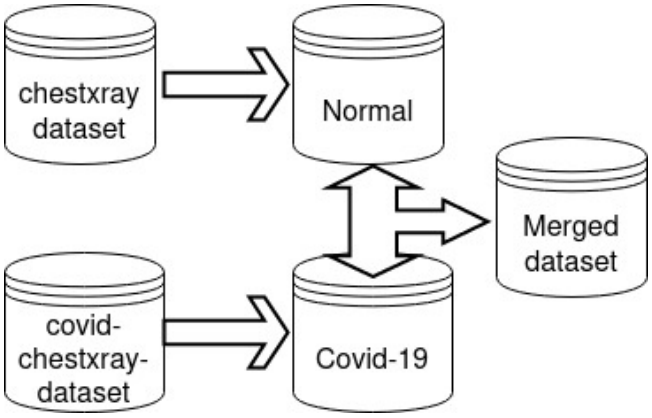


Figure 3.5: Dataset collection diagram

This initial step is followed by several treatments performed on these images to improve accuracy and data extraction. The next step applied a filtering and cleaning process to the thoracic medical images of COVID-19 patients. Only frontal radiological images confirmed by polymerase chain reaction (PCR) were retained. For the medical images of patients without lung diseases, we took a clear, high-quality dataset.

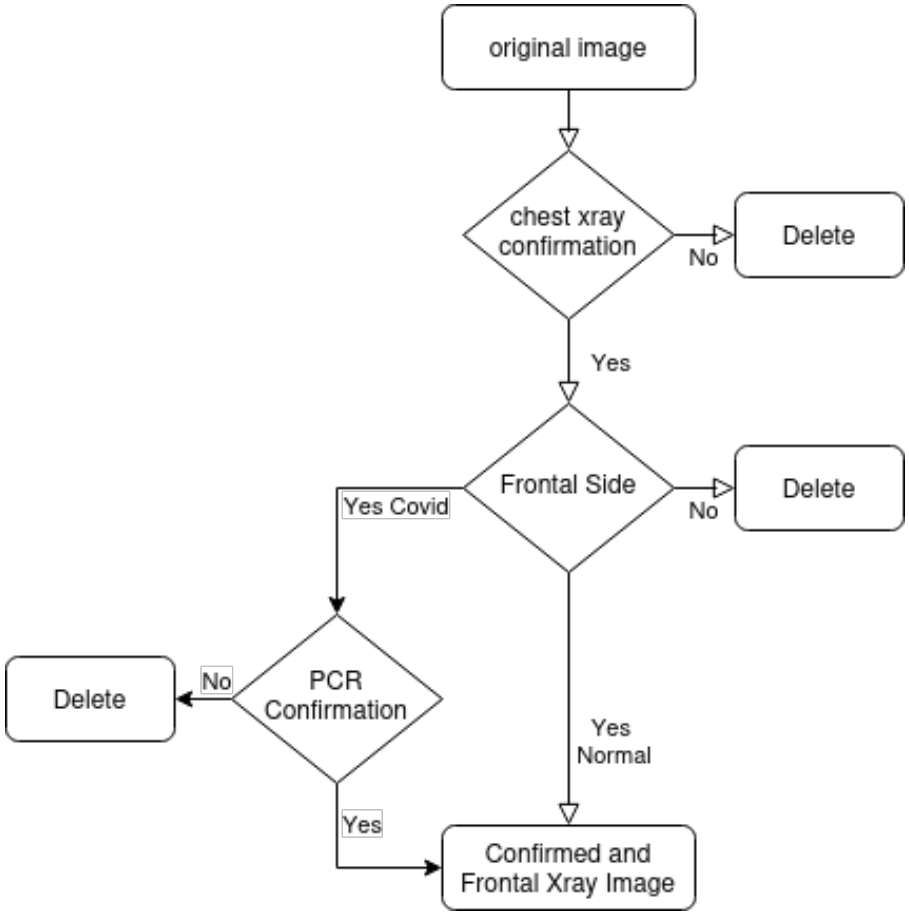


Figure 3.6: Diagram of verification/confirmation of the image's quality

Therefore, a new reliable and confirmed dataset was created. It contains two classes: one

named “Covid” for chest radiological images of COVID-19 patients, and the other named “Normal” for chest radiological images of normal patients. Both classes contain approximately 200 non-augmented images.

At the end we had a dataset containing hundreds of images from every continent and many countries to give a rich and divers samples of confirmed ill/infected persons.
To validate the results, we tried to access more data from hospitals but due to privacy and confidentiality issue. we didn’t had much data to train on,that’s why we opted for the open access data from internet, and inspected every single image and kept only PCR verified x-ray images, and at the end we generalized on five external datasets from unseen images and our model gave the best True positive values and the less miss classification rate.

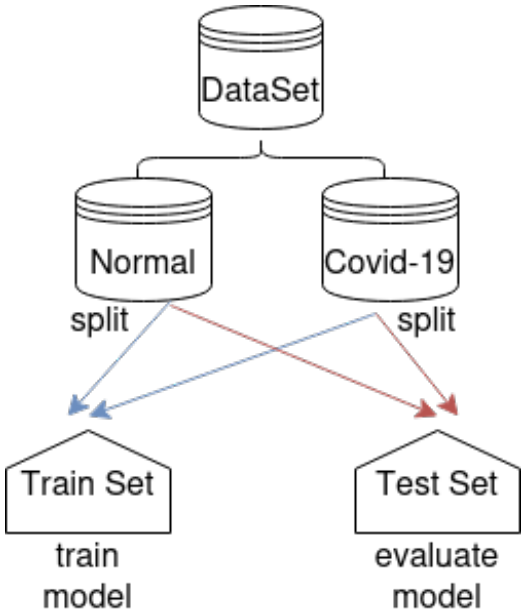


Figure 3.7: Global diagram of our dataset

The ’Covid’ class actually comprises 195 non-augmented images of PCR-confirmed chest radiological images from COVID-19 patients, while the ’Normal’ class includes 200 non-augmented images of chest radiological images from normal patients, we took 75% from every class for the train set and 25% for the test set, at the end we got 296 images (75%) (296 = 150 normal + 146 covid19) for training. And 99 images (25%) (99= 50 normal + 49 covid19) for the test set.

To reduce the dependencies of the model with the training data, a data augmentation process is used on the train set to increase the amount of data by adding slightly modified copies of already existing data. When trained, the model can see new variations of data at each epoch, and thus acts as a controller to avoid overlearning [169]. We can obtain this augmented data from the original images by applying simple geometric transformations, such as normalization, rotations, scale changes (zoom in/out), and horizontal flipping (Figure 3.20).

It is worth noting that data augmentation techniques were exclusively applied to the training set to enhance the generalization ability of our model. We took stringent measures to ensure that the test set remained entirely separate from any augmentation processes. This approach

was adopted to prevent any potential data breach and to maintain the test set's independence, thus upholding the robustness of our experimental evaluation.

3.4.4 X-ray images

An **X-rays**, or, much less commonly, **X-radiation** discovered by the german scientist *Wilhelm Conrad Röntgen* (November 8, 1895), is a penetrating form of high-energy electromagnetic radiation, where most wavelength frequencies are in the range 30 petahertz to 30 exahertz ($30x10^{15}$ Hz to $30x10^{18}$ Hz), shorter than those of **UV** (***UltraViolet***) **rays** and typically longer than those of ***gamma rays***.

X-ray images uses high-energy beams (electrons) to obtain pixel intensities of the human body, to form a gray scale image where the tissue density is indicated by shades of gray called Hounsfield scale (Figure 3.8). X-rays pass through human body tissues and hits a detector on the other side. Soft tissues absorb less radiation than dense tissues. A dense tissue (i.e. bones) will absorb more radiation than soft tissues (i.e. fat). When X-rays are not absorbed from the body (i.e. in the air region inside the lungs) and reach the detector we see them as black. On the opposite, dense tissues are depicted as white.

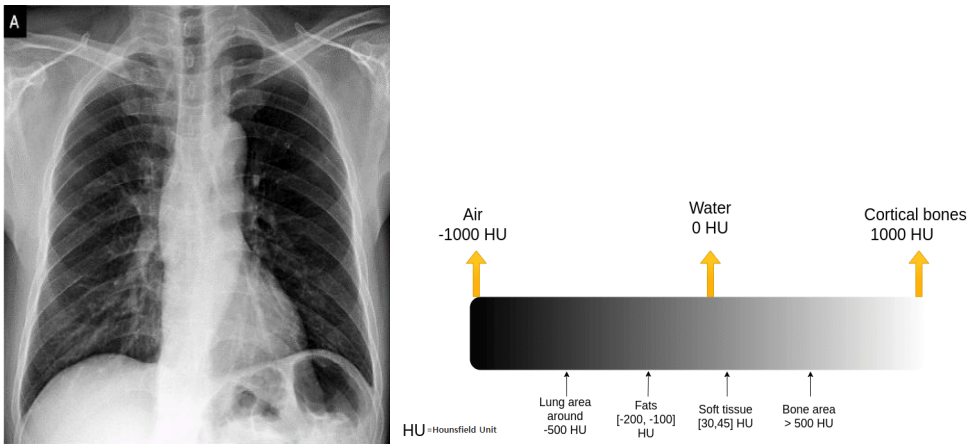


Figure 3.8: *The Hounsfield scale.*

The Hounsfield scale is a quantitative scale for describing radio density in medical X-ray images and provides an accurate density for the type of tissue Table 3.4. On the Hounsfield scale, air is represented by a value of -1000 (black on the grey scale) and bone between +700 (cancellous bone) to +1000 (dense bone) (white on the grey scale). As bones are much denser than surrounding soft tissues, they show up very clearly, as can be seen in Figure 3.8.

Tableau 3.4: X-ray image quality assessment: low versus high contrast resolution

Hounsfield units	Tissue
1000	Bone, calcium, metal
100 to 600	Iodinated CT contrast
30 to 500	Punctate calcifications
60 to 100	Intracranial hemorrhage
35 to 45	Gray matter
20 to 30	White matter
20 to 40	Muscle, soft tissue
0	Water
-30 to -70	Fat
-400 to -600	Lung Area
-1000	Air

3.4.5 Preprocessing

X-ray images may suffer from quality issues like underexposure, overexposure, motion blur, grid artifacts, positioning errors, metallic artifacts, diffusion effects, and ghosting/double images. These problems can reduce detail and accuracy in the images. X-ray images are inherently grayscale, meaning they represent variations in tissue density without using true color. Bones, which absorb more X-rays, appear lighter, while softer tissues, which allow more X-rays to pass through, appear darker. Achieving proper gray balance in X-ray images is essential for accurately conveying subtle differences in tissue density and pathology. While color is not used directly, achieving balance in the grayscale tones ensures accurate representation and interpretation of the image. Although X-rays are not visible light, the concept of "lighting" extends to the overall exposure conditions and imaging parameters in radiography. This indirectly influences the quality of X-ray images. Color balance is crucial in image processing to ensure accurate and consistent representation of colors. It's worth noting that while color balance might not be directly applicable to grayscale X-ray images, the principles of achieving a balanced and accurate representation of tones (gray balance) remain crucial in enhancing visual interpretation by indirectly affecting factors such as contrast and brightness.

Color/gray balance in X-ray images is a fundamental aspect of image quality and diagnostic precision. Achieving the right balance of tones ensures that model can effectively classify the images, leading to improved patient care and more accurate medical diagnoses.

3.4.5.1 Simplest Color Balance

The biggest problem in most of x-ray images is that their colors are not Enhanced, or not balanced (are said to have a *color cast*, since everything in the image appears to have been shifted towards one color or another (in the gray scale canal). Color balancing may be thought

in terms of removing this color cast.

When it comes to medical imaging and xray images, the effects of light can significantly impact the accuracy and reliability of the detection process.

We had several options to enhance the light and color cast, we have chosen the Simplest Color balance after several attempts with many light and color enhancement algorithms. and (the simplicity, the speed, and the adequacy of the used kind of data) of this algorithm were the most important parameters for our choice.

The X-ray images are typically grayscale, and color balance adjustments are primarily used for color images. The reason why we applied the simplest color balance adjustments to grayscale images like X-rays is that dealing with one channel (intensity/brightness) or three channels (chrominance) of each pixel leads to the same effect which is gray/color cast reduction.

in our case even if it seems abnormal to say that we are correcting color, when manipulating one channel we are correcting the light from the grayscale space or Y, which is considered as color but in reality, it represents the light (Luma) information, and in the x-ray situation we are dealing with light more than a color, but it still considered as a color space with one channel. if you take a look at the grayscale formula (3.1 or 3.2) and Y (luma = luminance) formula (3.3 or 3.4) they are almost the same:

1- for the Gray Scale, we have two options:
Average Color Values:

$$Gray = (Red + Green + Blue)/3 \tag{3.1}$$

Or, ITU BT.709 [170] or ITU BT.601 [171]:

$$Gray = 0.299 * Red + 0.587 * Green + 0.114 * Blue \tag{3.2}$$

2- for the Y Scale, we have two options:
Photometric/digital ITU BT.709 [170]:

$$Y = 0.2126 * Red + 0.7152 * Green + 0.0722 * Blue \tag{3.3}$$

Digital ITU BT.601 [171] (gives more weight to the R and B components):

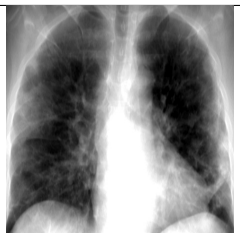
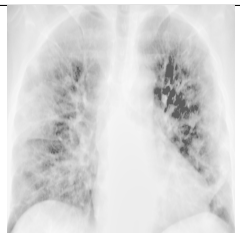
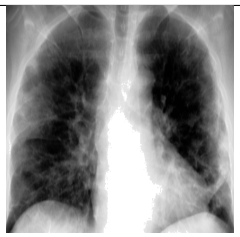
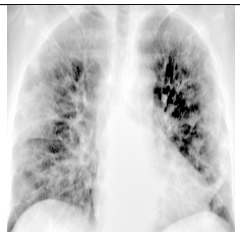
$$Y = 0.299 * Red + 0.587 * Green + 0.114 * Blue \tag{3.4}$$

Simplest Color Balance is an image processing technique that aims to correct color imbalances in an image. It adjusts the color channels (usually Red, Green, and Blue or ***gray scale like in the case of x-ray images***) of the image to make the colors appear more accurate and balanced. The process involves applying scaling factor to the channel to achieve the desired balance.

Simplest Color Balance can be particularly useful when an image has gray cast or an unwanted light tint. It helps bring the tone back to a more natural and pleasing state by equalizing the intensity of the gray channel.

Adjusting the overall intensity of the colors in the grayscale is what color balancing and contrast enhancement are all about. The goal of this adjustment is to correct underexposed images, or images taken under artificial lights or special lights, such as x-rays; thus, the general method is sometimes referred to as gray balance, neutral balance, or white balance. Numerous methods have been proposed in the state-of-the-art to address color balancing. Table 3.5 provides a qualitative analysis of the most well-known image enhancement algorithms and their impact on medical X-ray images.

Tableau 3.5: Visual comparison of the most famous color/light enhancement algorithms

			
Simplest CB	MSRCR	HE (unnatural zones)	White balance
			
Automated MSRCR	MSRCP	CLAHE	DHE

3.4.5.2 Why Simplest CB:

We opted to utilize the Simplest Color Balance (SCB) in this study for the following reasons:

- Although this technique does not always enhance the image, it consistently improves readability.
- It is highly efficient in terms of time and memory consumption. We implemented a histogram sorting and clipping algorithm, which is less complex, faster, and simpler compared to the matrix-based approach of sorting N pixel values, requiring $O(N \log(N))$ operations and more memory due to the temporary copy of all pixels.
- SCB does not significantly impact well-saturated images, preserving their quality and maintaining the saturation and light balance. However, it effectively enhances under saturated and poorly illuminated images.
- Particularly when applied to medical X-ray images, SCB surpasses the performance of other well-known light enhancement algorithms.

3.4.5.3 Algorithm

The saturation extremas $S_{\min}$ and $S_{\max}$ can be calculated after a sorting step. Therefore, we opted for a histogram-based sorting method due to the following reasons:

A classic sorting method computes $S_{\min}$ and $S_{\max}$ by sorting the N pixel values and selecting the quantiles from the sorted array. However, sorting the N pixel values necessitates $O(N \log(N))$ operations and the creation of a temporary copy of these N pixels. Thus, choosing a more efficient implementation becomes imperative.

By utilizing a histogram-based sorting method, we achieve a faster complexity of $O(N)$ and require less memory, with a complexity of $O(255)$ compared to $O(N)$.

a) **Build a cumulative histogram of the pixel values**: Count the cumulative number of observations in all of the bins up to the last bin.

b) **Pick the quantiles from the histogram**: $S_{\min}$ is the lowest histogram label with a value higher than $N \times P/100$, and the number of pixels with values lower than $S_{\min}$ is at most $N \times P/100$. $S_{\max}$ is the label immediately following the highest histogram label with a value lower than or equal to $N \times (1 - P/100)$, and the number of pixels with values higher than $S_{\max}$ is at most $N \times P/100$.

c) **Saturate the pixels**: Apply the threshold.

d) **Affine transform**: Scale the image to min = 0 and max = 255, which are the limits of the gray scale $[0, 255]$, with an affine transformation of the pixel values by the function f such that $f(x) = \left(\frac{(x - S_{\min}) \times (\max - \min)}{S_{\max} - S_{\min}}\right) + \min$.

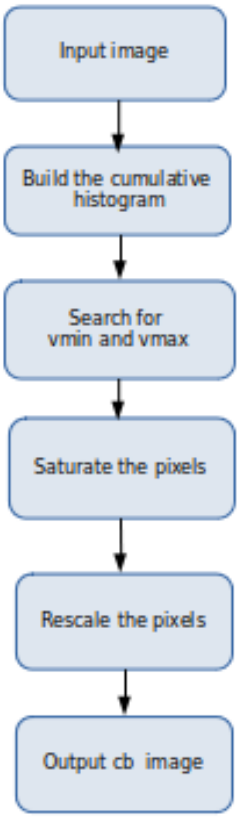


Figure 3.9: The simplest color balance algorithm steps

The following Algorithm 1 applies to images with pixel values in the 8 bit integer space (min = 0, max = 255) with one color channel only.

In Algorithm 1 $image[i]$ are the pixel values, N is the number of pixels, *histo* is an array of 256.

Algorithm 1 Simplest Color Balance algorithm

Require:

$image$:Original image, P :clipping percentage;

Ensure:

enhanced image;

//build the cumulative histogram

1: **for** $i \leftarrow 0$ to $N-1$ **do**
2: $histo[image(i)] := histo[image(i)] + 1$
3: **end for**
4: **for** $i \leftarrow 1$ to 255 **do**
5: $histo[i] := histo[i] + histo[i-1]$
6: **end for**

// search S_{min} and S_{max}

7: $S_{min} := 0$
8: $S_{max} := 255$
9: **while** $histo[S_{min}+1] \leqslant (N \times P/100)$ **do**
10: $S_{min} := S_{min} + 1$
11: **end while**
12: **while** $histo[S_{max}-1] > N \times (1 - P/100)$ **do**
13: $S_{max} := S_{max} - 1$
14: **end while**
15: **if** $S_{max} < 254$ **then**
16: $S_{max} := S_{max} + 1$
17: **end if**

// saturate the pixels (truncation)

18: **for** $i \leftarrow 0$ to $N-1$ **do**
19: **if** $image[i] > S_{max}$ **then**
20: $image[i] := S_{max}$
21: **end if**
22: **if** $image[i] < S_{min}$ **then**
23: $image[i] := S_{min}$
24: **end if**
25: **end for**

// re-scale the pixels (normalization)

26: **for** $i \leftarrow 0$ to $N-1$ **do**
27: $image[i] := (image[i] - S_{min}) \times 255/(S_{max} - S_{min})$
28: **end for**

The percent parameter is expressed in natural language (where number 1 represents 1%), thus it undergoes a division by 100 to express it as a fraction of 1, which is easier to work with (eliminating the need for normalization at each computation). The division by 2 is to obtain its

half, with one half used to adjust the darks and the other half to adjust the brights. Combining the two divisions results in a division by 200.

The underlying assumption of using this algorithm is that the highest values of chrominance observed in the image correspond to white, while the lowest values correspond to obscurity. The concept is based on the idea that in a well-balanced X-ray image, the brightest color should represent white (bone) and the darkest color should represent black (air). Therefore, we aim to remove the color cast from the image by scaling the histogram to span the complete 0-255 scale through an affine transform $ax + b$.

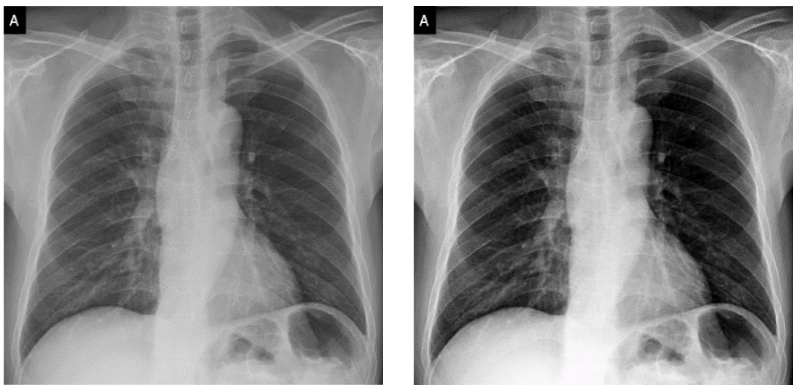
However, while balancing an image affects not only the neutrals but also other colors, the challenge lies in balancing these colors while ensuring they appear correct or pleasing at the same time.

Since many images contain a few aberrant pixels that already occupy the 0 and 255 values, the proposed solution is to saturate or "truncate" a small percentage $P1\%$ of the pixels with the highest values to 255 and a small percentage $P2\%$ of the pixels with the lowest values to 0. This approach helps avoid normalization at each computation step and makes the process faster (less time-consuming). To maintain efficiency and consistency, we choose the same percentage $P\%$ for both dark and white intensities ($P = P1 = P2$). The percent parameter is expressed in natural language (for example, $P = 3$ means that this balance will saturate $N \times \frac{3}{100}$ pixels at the black (air) intensities and will saturate at most $N \times (1 - \frac{3}{100})$ pixels at the white (bones) intensities).

To prevent the creation of flat white or black regions that may appear unnatural, it is crucial to keep the adjustable parameter of the saturation level as small as possible [172] (typically 0.01 or 1%). Finally, after saturation, we apply the affine transformation.

In his paper, "Comparison of the accuracy of different white-balancing options as quantified by their color constancy," *S. Viggiano* [173] observed that white, gray, or color balancing tended to result in less color inconsistency, implying reduced distortion of colors. In other words, achieving the correct color balance during the capture of raw image data proves advantageous as it tends to minimize chromatic color distortion.

The overall algorithm for the Simplest Color Balance (SCB) is explained in Algorithm 1, and the main result is illustrated in Figure 3.10.



Original Enhanced

Figure 3.10: Results of the proposed Simplest Color Balance applied to x-ray image

Incorporating the X-ray image in its current state yields poor results in both training and detection. The quality of the image must be enhanced, depending on the specific field in which it will be utilized. As an initial pre-processing step, a color balance algorithm known as Simplest Color Balance is applied to accentuate the finest details. Additionally, it's worth noting that the X-ray image may contain information about other parts of the body unrelated to the COVID-19 disease.

Consequently, a second preprocessing step is employed to extract the region of interest (ROI). Figure 3.11 illustrates all the preprocessing steps utilized.

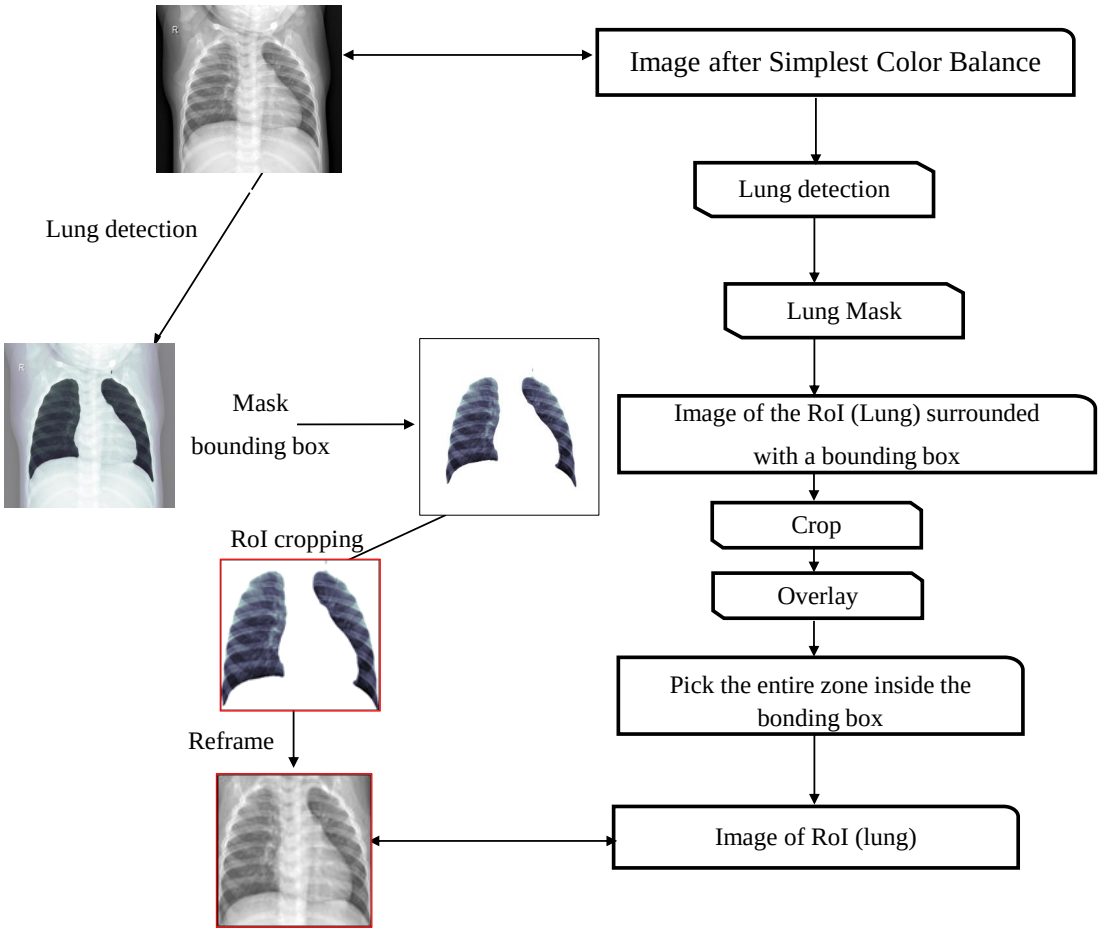


Figure 3.11: Preprocessing steps

3.4.5.4 Test processeing step

After completing the enhancement process, we observed that certain x-ray images were already of good quality. Although the SCB algorithm did not alter these images, the implementation of the algorithm resulted in a loss of time. As a solution, we introduced an additional step to determine whether applying the color balance adjustment was necessary.

test low contrast:

The "test low contrast" algorithm serves to ascertain whether an image possesses low contrast. Its core principle involves assessing the standard deviation of pixel intensities within the image against a predefined threshold. If the standard deviation falls below this threshold, the algorithm identifies the image as having low contrast.

The algorithm's steps are summarized as follows:

(1) Convert the image to grayscale.

(2) Calculate the mean intensity of the grayscale image.

(3) Determine the standard deviation of intensity values in the grayscale image.

(4) Compute the contrast fraction, defined as the ratio of the standard deviation to the mean intensity.

(5) Identify the image as having low contrast if the contrast fraction is below a specified threshold.

We have set the threshold value used in step 5 to 0.4, although this can vary depending on the specific application.

We suggest employing the "test low contrast" algorithm as a preliminary step to determine whether to use the Simplest Color Balance algorithm or proceed directly to the Lung Segmentation step.

If the "test low contrast" algorithm indicates that the image has sufficient contrast, then we apply the Simplest Color Balance algorithm to correct the image's grayscale color and details, resulting in more natural and visually appealing results.

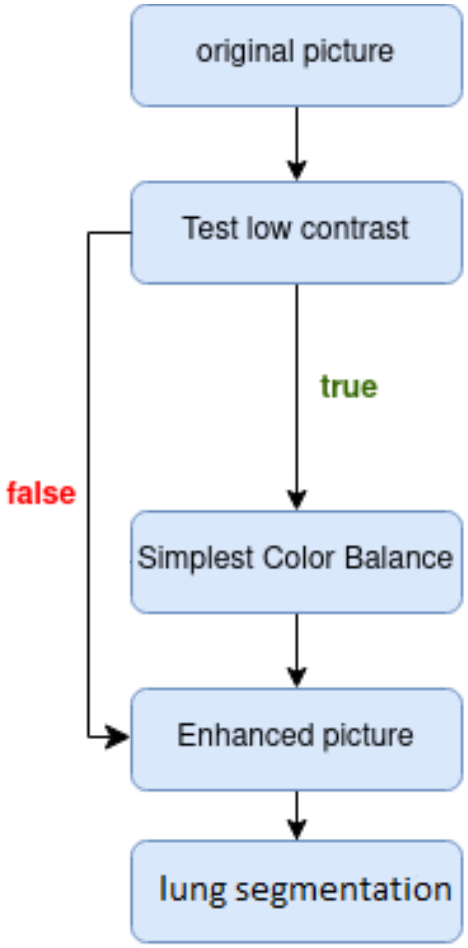


Figure 3.12: test low contrast for SCB enhancement

3.4.5.5 Lung segmentation using UNet

Working on the entire X-ray image may affect the prediction result in the future. This is because the model learned unimportant features like the shoulders, neck, and abdomen. Furthermore, the additional data causes the system to take longer to train our model.

As a result, for a good and efficient result, it is necessary to work only on the lungs and not on the complete X-ray image, because if we provide complete images without specifying the areas of interest in which the model must train, causing misleading training , and this

could affect the results of the prediction in the future and the model learn about unwanted and unimportant features from liver, shoulders, neck, and abdomen...etc.

Automatic medical image segmentation is a crucial topic in the medical domain and successively a critical counterpart in the computer-aided diagnosis paradigm. To detect the region of interest (lung area), we will implement a pixel-by-pixel (or pixel wise) segmentation strategy.

The most obvious solution for semantic segmentation problems is to use segmenting lungs method on chest X-Ray images inspired from an encoder-decoder method based on UNet architecture, due to its flexibility, optimized modular design, and success in all medical image modalities, developed mainly for biomedical image segmentation, was created by [***Olaf Ronneberger, Philipp Fischer and Thomas Brox*** in 2015 [174], "**UNet**: *convolution networks for biomedical image segmentation*"], Which is an improvement and development of convolutional neural network architecture (called "fully convolutional network" [***Evan Shelhamer, Jonathan Long and Trevor Darrell*** (2014) [175], "Fully convolutional networks for semantic segmentation"]), its architecture has been modified and extended to work with fewer training images and to allow more accurate and fast segmentation of high resolution images with fewer training images(512 * 512 image takes less than a second on a recent GPU).

We proposed to use UNet architecture designed for semantic segmentation tasks, where the goal is to assign each pixel in an input image to a particular class or category.

The main idea consists of a contracting path with repeated application of convolutions, each followed by a rectified linear unit (ReLU) and a maximum pooling operation, to capture context information while spatial information is reduced and feature information is increased. And a symmetric expanding (resampling) path that combines geographic and spatial feature information through a sequence of upward convolutions and concatenations with high resolution features from the contracting path, which allows the network to propagate context information to higher resolution layers to enable precise mask localization.

The main idea is to use two paths, as shown in Figure 3.13. This is called a contracting path which involve a series of convolutional layers with ReLU activation followed by max-pooling operations. This helps to reduce spatial dimensions while increasing feature information, allowing the network to capture context information effectively. On the other hand, a symmetric expanding (resampling) path, or decoder, involves upsampling operations to restore the spatial resolution. It combines both geometric and spatial features through a sequence of upward convolutions and concatenations with high resolution features from the contracting path, which allows the network to propagate context information to higher resolution layers to enable precise mask localization/segmentation.

A final convolutional layer with a ***softmax*** activation function was applied to the model output for the final level. It generates a segmentation probability map using the ***Adam*** optimization criterion and a ***learning rate*** of *0.0005*.

A detailed breakdown of how the UNet works:

(a) **Encoder (Contracting Path):**

- The architecture begins with a series of convolutional layers, known as the contracting path or encoder.
- Each convolutional layer is followed by a rectified linear unit (ReLU) activation function, introducing non-linearity.
- Max-pooling operations are applied to reduce spatial dimensions, capturing hierarchical features in a coarse-grained manner.
- The contracting path progressively downsamples the input image, extracting high-level abstract features.

(b) **Bottleneck:**

- At the end of the contracting path, a bottleneck layer is formed. This layer captures the most abstract features of the input image, representing a compressed and abstracted version of the original data.

(c) **Decoder (Expansive Path):**

- The expansive path or decoder follows the bottleneck layer. It consists of a series of upsampling operations and convolutional layers.
- Each upsampling operation increases the spatial resolution, allowing the network to capture fine details.
- Skip connections are introduced by concatenating feature maps from the corresponding level in the contracting path. This helps in preserving spatial information lost during downsampling.

(d) **Final Layer:**

- The final layer of the network is a convolutional layer with a kernel size of 1x1, which acts as a pixel-wise classifier.
- A sigmoid activation function is applied to produce pixel-wise probabilities for each class in binary segmentation.

(e) **Loss Function:**

- Common loss functions for semantic segmentation tasks with UNet include the dice coefficient loss.
- The model is trained using backpropagation and ***Adam*** optimization criterion and a ***learning rate*** of *0.0005*, minimizing the loss function.

Key Features:

- Skip Connections: The skip connections between the encoder and decoder help the network preserve spatial information and gradients during training, enabling the model to better capture details.

- U-Shape Architecture: The U-shape design, with a contracting path followed by an expansive path, inspired the name "UNet."

NOTE

- Some researchers prefer to use a batch normalization layer in between the convolution layer and the ReLU activation function. The batch normalization reduces internal covariance shift and makes the network more stable while training.
- The dropout is also used sometime after the ReLU activation function. It forces the network to learn a different representation by dropping out (ignoring) some randomly selected neurons. It helps the network to become less dependent upon certain neurons. This in turn helps the network to better generalize and prevent it from overfitting.

UNet efficiently captures both local and global context in images through its contracting and expansive paths, making it well-suited for our case where precise localization and segmentation of ROI which is the lungs is crucial.

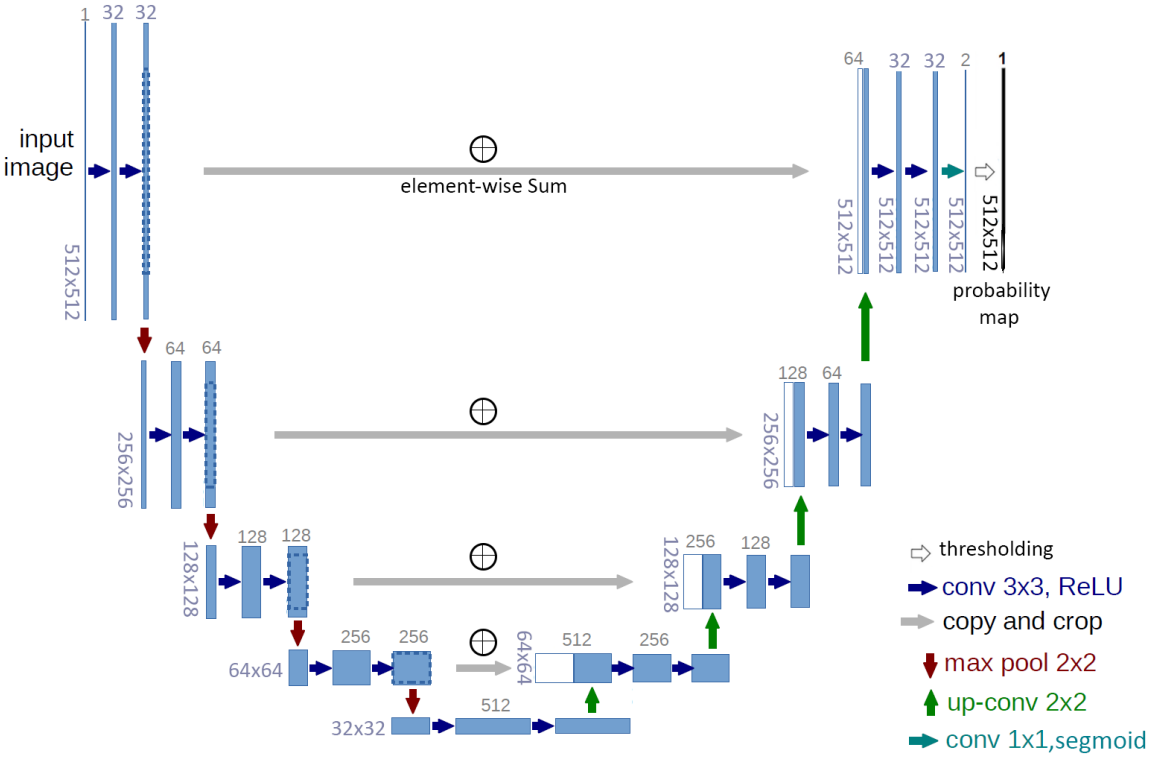


Figure 3.13: Detailed architecture of the used UNet network

Figure 3.13 depicts the architecture of the used UNet. A convolution with a kernel size of 3x3 is represented by each blue right arrow. The numbers beside and above each blue box represent the size (height and width) and number of feature maps. The red (down) arrows represent max-pooling layers with kernel sizes of 2x2 and strides of 2x2, which reduce the size of their input by half. The white boxes are copies of feature maps.

- **Training UNet:**

The model was trained using two publicly available chest X-ray datasets for computer-aided screening of pulmonary diseases [176]. The first was collected as part of the Montgomery County Department of Health and Human Services tuberculosis control program. The dataset contains 138 posterior-anterior x-rays, 80 of which are normal and 58 of which are abnormal with tuberculosis manifestations. The second dataset was developed in collaboration with Shenzhen No.3 People's Hospital, Guangdong Medical College, and the National Library of Medicine, National Institutes of Health, Bethesda, MD, USA. It contains 336 tuberculosis cases and 326 normal cases collected from outpatient clinics.

- **DataSets Description:**

Montgomery County chest X-ray set : The Montgomery County 1 set has been collected in collaboration with the Department of Health and Human Services, Montgomery County, Maryland, USA. The set contains 138 frontal chest X-rays from Montgomery County's Tuberculosis screening program, of which 80 are normal cases and 58 are cases with manifestations of tuberculosis. The X-rays were captured with a Eureka stationary X-ray machine, and are provided in grayscale *.PNG format images. The size of the X-rays is either 4,0204,892 or 4,8924,020 pixels.

All image file names follow the same template: MCUCXR_####_X.png, where #### represents a 4-digit non-sequential numerical identifier, and X is either 0 for a normal X-ray or 1 for an abnormal X-ray.

Manual lung segmentations are also available (For each X-ray, the corresponding binary lung mask separately for the left and right lung are saved).

China Set - The Shenzhen set - Chest X-ray Database: The Shenzhen set for Tuberculosis 2 is created by the National Library of Medicine, Maryland, USA in collaboration with Shenzhen No.3 People's Hospital, Guangdong Medical College, Shenzhen, China. The Chest X-rays are from out-patient clinics, and were captured as part of the daily routine using Philips DR Digital Diagnose systems. The set contains 662 frontal chest X-rays, of which 326 are normal cases and 336 are cases with manifestations of tuberculosis, including pediatric X-rays (AP). The X-rays are provided in PNG format. Their size can vary but is approximately 3000 3000 pixels.

Image file names are coded as CHNCXR_#####_0/1.png as we can see in Figure 3.14, where '0' represents the normal and '1' represents the abnormal lung.

Figure 3.14: some samples from The Shenzhen dataset

The x-ray images and corresponding masks from the two datasets were resized to *512x512* for training by a batch of 4 images during more than 50 epochs. While visualizing the training figures Fig 3.15 we noticed that after 40 epoch network stops to improve the validation score and the network began to overfit, and the weights with best validation scores were selected and saved.

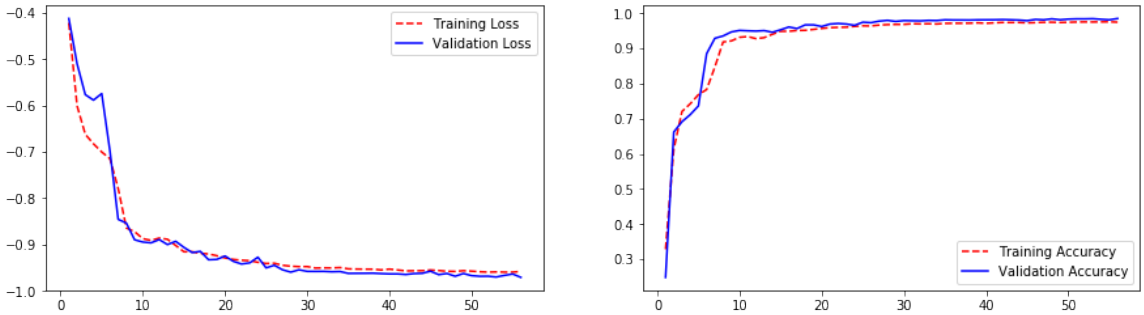


Figure 3.15: visualization of Accuracy/Loss evolution while training UNet

So, for more details about the training step of our UNet we used the augmented two datasets, and hyperparametrs for Tuning the UNet model as in the following nomenclature:

(a) Augmentation:

- rotation range=0.2,
- width shift range=0.05
- height shift range=0.05
- shear range=0.05
- zoom range=0.05
- horizontal flip=True

- fill mode='nearest'

(b) Model parameters:

- input size=(512,512,1)
- optimizer=Adam(lr=1e-5),
- loss=dice coefficient loss,
- metrics=[dice coefficient, 'binary accuracy']
- save best only=True
- BATCH SIZE=4
- steps per epoch=len(train files) / BATCH SIZE
- EPOCHS=56

The results after applying the weights on Xray images can be seen in figure 3.16.

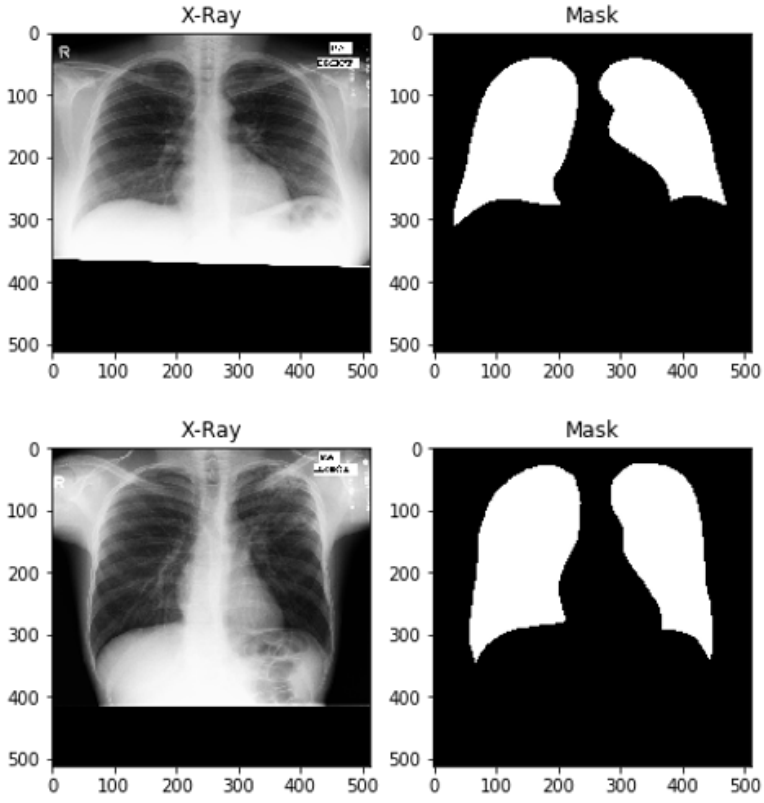


Figure 3.16: Unet segmentation result

3.4.5.6 RoI Cropping

After the initial lung segmentation, post-processing steps were used to determine the final cropping area using the segmented mask.

The first choice was to take the mask and crop only the detected area by superposing the mask on the original input image, but we noticed that in some cases the lung detection doesn't recognize ill tissues as lung (mistakenly classified as non-lung) so those tissues are deprecated,

To address this, we opted for a bounding box approach, collecting all the area of segmented lung and surrounding zones by reducing the region of interest to take the minimum of non-interesting information.

To do that, all connected regions with eight connectivity pixels were identified. Since the two largest regions are most likely the left and right lungs, an initial bounding box around the biggest two regions was selected. Thereafter, To prevent the inadvertent removal of useful information, a small safety distance was added to the initial bounding box at the top, left, right and bottom of the bounding box, ensuring a more accurate representation of the lung region while preserving relevant details, the result is each image was cropped to its individual bounding box.

This approach is aimed at refining the segmentation process and minimizing the risk of excluding important data.

Figure 3.17 illustrates the details of the Lung segmentation and ROI cropping on chest X-ray images, *From up-left to down-right. (1) The original chest X-ray image was processed by UNET (2) to generate (3) the lung mask segmentation, (4) presents the calculated bounding box around the two largest connected regions. In (5) the safety area of the bounding box. (6) shows the final cropped image.*

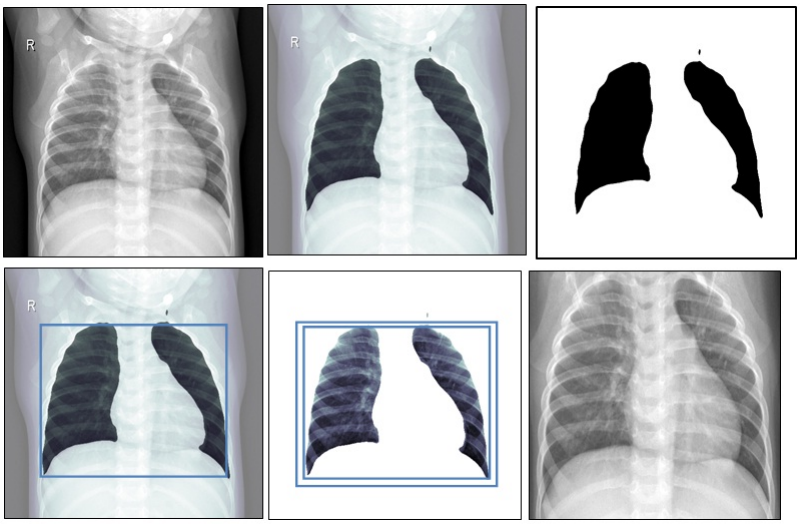


Figure 3.17: Overview of the Lung segmentation on chest Xray images method.

Figure 3.18 depicts the architecture of the used UNet as it is shown in the framework of tensorflow.

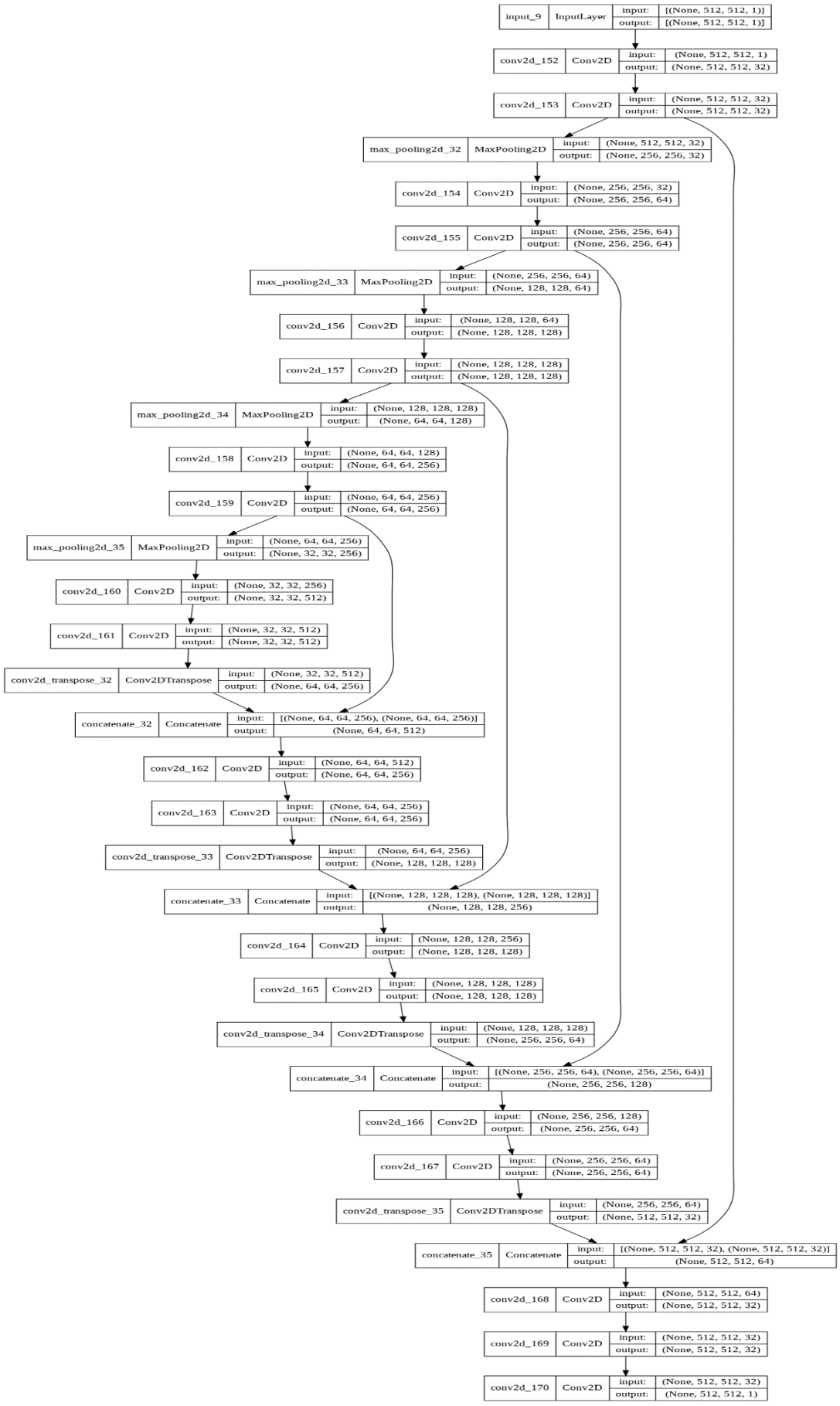


Figure 3.18: the architecture of the used UNet as it is shown in the framework of tensorflow.

3.4.6 Data augmentation

As a general rule of thumb, the big advice to follow is to have between 1000~5000 images per class when you want to train a deep neural network algorithm, and to overcome the challenge of overfitting.

most of machine learning models, and especially neural networks, can require quite a bit of training data - but in some situations we don't have very much training data in the first place, like in our case where the world was surprised by the disease, and the leak of data.

We cannot expect to train a CNN on a small amount of data and then expect it to generalize to data it was never trained on and has never seen before.

one the most useful and practical methods to overcome this problem is oversampling or data augmentation.

So to win a big effort and time and give CNN more data to learn from. we want to ensure that our network, when trained, sees new variations of our data at each and every epoch, we can obtain that augmented data from the original images by applying simple geometric transformations, such :

- Translation (the image is moved along the X, Y directions).
- Rotations
- Scale changes (zoom in/out)
- Horizontal flipping

Applying a (small) amount of the transformations to an input image will change its appearance slightly, but it does not change the class label.

In order to reduce the dependence on training data preparation and build more accurate deep learning models faster. Data augmentation is an approach to generate new variations of data, used to increase the amount of data by adding slightly modified copies of already existing data, to save a lot of effort and time and give the CNN more data to learn. We want to ensure that our network, when trained, sees new variations of our data in each epoch, It acts as a controller and helps reduce overlearning when training a machine learning model. It is closely related to oversampling in data analysis.

3.4.6.1 why we have to use data augmentation

Data augmentation is crucial technique in machine learning, and specifically in the context of COVID-19 detection, for enhancing the training of the model by providing diverse representations of cases, improving *robustness to variability*, and *mitigating overfitting*.

In our context of limited data availability and the high costs associated with collecting and annotating medical images, augmentation techniques offer a cost-effective solution by maximizing the utility of existing datasets.

Furthermore, the augmentation process ensures that the model is exposed to a broad spectrum of scenarios, leading to improved generalization and performance in real-world settings. By addressing the challenges of diverse patient populations and varying imaging conditions, data augmentation becomes an indispensable component in the development of accurate and reliable COVID-19 detection models.

In summary, data augmentation is a fundamental technique that addresses challenges related to limited data, variability, and overfitting, ultimately enhancing the robustness and generalization capabilities of a COVID-19 detection model :

(a) **Increased Data Diversity:** Data augmentation involves creating new training examples by applying various transformations (such as rotation, scaling, and flipping) to the existing dataset. This process introduces diversity, ensuring that the model sees a broader range of scenarios and variations in COVID-19 cases. This diversity is crucial for the model to generalize well to unseen data.

(b) **Improved Robustness:** By exposing the model to augmented data, it becomes more robust to variations in imaging conditions, such as changes in lighting, angles, and perspectives. This helps the model perform well under different real-world situations, making it more reliable for practical applications.

(c) **Mitigation of Overfitting:** Data augmentation introduces randomness during training, preventing the model from memorizing specific patterns in the training set (overfitting). Overfitting occurs when a model learns noise in the training data rather than capturing the underlying patterns. Augmentation helps the model generalize better to new, unseen data.

(d) **Optimal Use of Limited Data:** In many medical domains, including COVID-19 detection, obtaining a large labeled dataset is challenging due to privacy concerns and resource constraints. Data augmentation allows you to effectively increase the size of your dataset, making the most out of the available labeled examples.

(e) **Cost-Effectiveness:** Collecting and annotating medical images can be expensive and time-consuming. Augmenting existing data reduces the need for additional labeled samples, making the training process more cost-effective while still improving the model's performance.

(f) **Generalization to Diverse Patient Populations:** COVID-19 cases can manifest differently in diverse patient populations. Augmenting data ensures that the model is exposed to a wide range of patient characteristics, improving its ability to detect the disease across different demographics.

We employed a widely used data augmentation technique known as *in-place data augmentation* or *on-the-fly data augmentation*, which is applied during training. In this method, while

our model undergoes training, batches of original data are randomly transformed and then fed to the neural network.

This process involves three main steps:

Step 1: Inputting a batch of images.

Step 2: Applying a series of selected transformations to each image in the batch.

Step 3: Returning the randomly transformed batch to our model for training.

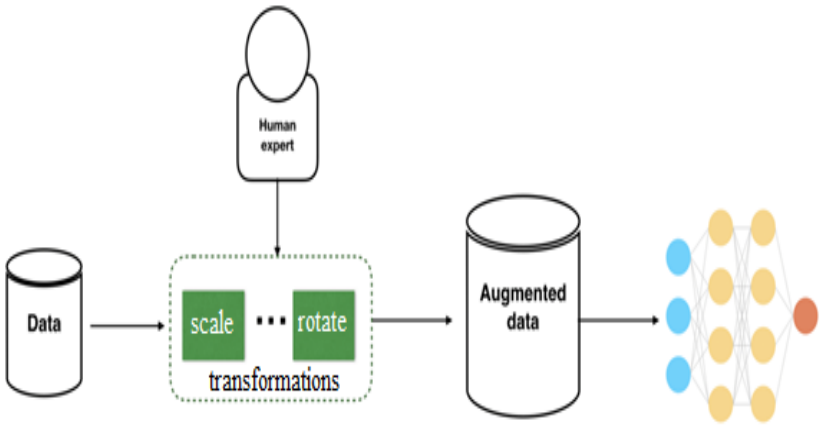


Figure 3.19: Transformation functions for In-place/on-the-fly data augmentation

We call this augmentaton "*in-place*" and "*on-the-fly*" data augmentation because this augmentation is done ***at training time*** (i.e., we are not generating these examples ahead of time/prior to training). It's similar to ***incremental learning***.

In short way to explain this, while our model is being trained, we randomly transform "intercepted" original data to encrease the quantity and the quality, and then return it to the neural network for training.

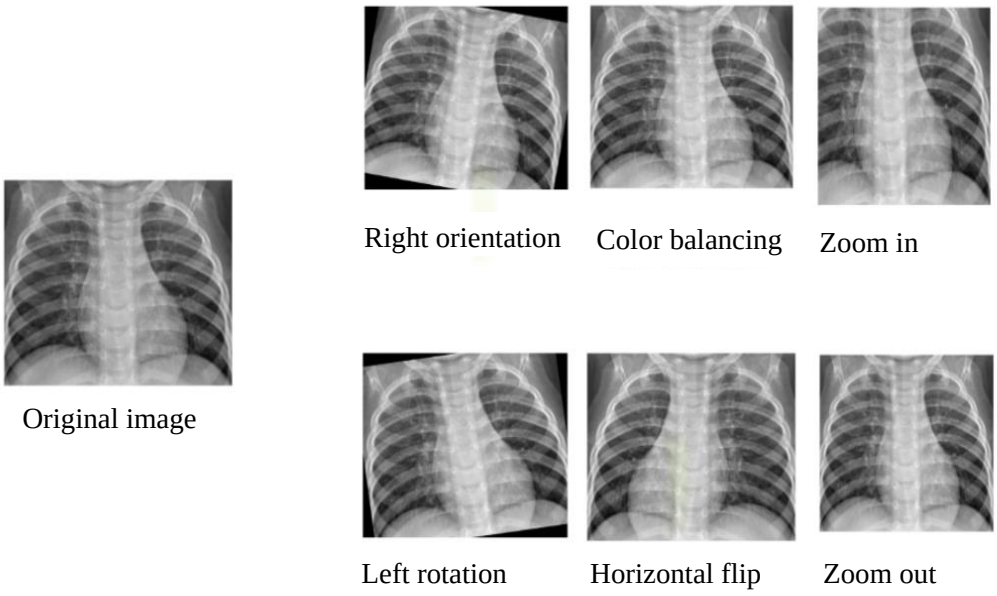


Figure 3.20: Image augmentation

Very important: We want to clarify that the test set is invisible to the training process, and data augmentation was exclusively applied to the training set to maintain the integrity of the test set and enhance the generalization ability of our model. We ensured that the test set remained unaffected by any augmentation processes, thereby preserving its independence and validity in assessing the model's performance.

3.5 X-ray Image Classification Model

In our case, the classification problem is a challenging machine learning issue, but still practical realistic one, caused by the leak of data. since a small-scale set of data collection can be extremely expensive or sometimes near-impossible (e.g. in medical imaging, and it's much more harder in the case of a pandemic), meanwhile having very few samples to learn and memorize from, the goal is we intend to reach/overcome the skill of being able to extract the most out of very little data, by find the best model that fits with this kind of information (x-rays), and to **improve the accuracy** and in the same time **minimize the loss function,** without using any weights or any custom feature engineering

In the latest stat of art, the accuracy score reached 98% by using modern deep learning techniques and huge datasets, in our case, the problem is much harder.

Most of researchers says "deep learning is only relevant when you have a huge amount of data". While not entirely incorrect, this is somewhat misleading. Certainly, deep learning requires the ability to learn features automatically from the data, which is generally only possible when lots of training data is available -especially for problems where the input samples are very high-dimensional and inter-classes and intra-classes are very similar, like x-ray images. However, convolutional neural networks are by design the best models available for most "perceptual" problems, even with very little data to learn from.

Training a CNN from scratch without using any weights on a small image dataset will still yield reasonable results, without the need for any custom feature engineering.

But what's more, deep learning models are by nature highly repurposable: you can take, an image classification model trained on a large-scale dataset then reuse it on a significantly different problem with only minor changes.

Specifically, in the case of computer vision, many pre-trained models (usually trained on the ImageNet dataset) are now publicly available for download and can be used to generate powerful vision models out of very little data, so we tried this option with a few knowen models pretrained.

based on the type/quantity of our images and in light of the problem of the lack of available examples and the importance and difficulty of dealing With the medical information based on that, we had to be careful to choose a method suitable for similar cases in terms of the lack of examples and the difficulty of differentiating between classes, as well as being able to generalize work on new examples with ease and with a very small error rates.

In this section, we present the used approach to build and train a robust convolutional

neural network model that allows us to classify COVID'19 disease in lungs detected in chest X-ray images; using only very few training samples per class.

To do that, and after the previous stages that we performed, which are very important for preparing the images to enhance the lighting of the images, and also to take into account only the important areas, which are the lung without other organs that may cause wrong and ambiguous learning. we had those options: (*train well known models in table 3.6 that has proven their effectiveness such as (InceptionV3, DenseNet, VGG16, VGG19, ResNet50), using the bottleneck features of a pre-trained network, fine-tuning the top layers of a pre-trained network by freezing all the first layers* or *training a* ***baseline*** *network from scratch)*:

- First, working on one of the models that has proven its effectiveness in other areas of object recognition systems such as: "InceptionV3, DenseNet, VGG16, VGG19, ResNet50" without using weights.
- Secondly, train these pre-trained models on ImageNet dataset by using the bottleneck features.
- third, Fine-tuning the last layers of thses pre-trained networks by freezing all the first layers.
- And fourth, the development of a personal model with the use of the largest number of parameters to reach the greatest effectiveness with the least number of examples from each class, in other word: "training a small network from scratch (**as a light weight baseline model**)".

Finally, in each of the cases, we tested a large number of hyperparameters to increase efficiency and reduce loss. These settings are learning rate, freezing layers, pretrained weights, changing out layer, and using early stopping to avoid overfitting and to match the lose function the type of problem to make the results better and better.

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
DenseNet121	33	75.0%	92.3%	8.1M	242	77.1	5.4
DenseNet201	80	77.3%	93.6%	20.2M	402	127.2	6.7

Tableau 3.6: The selected well known models that has proven their effectiveness

*The **top-1** and **top-5** accuracy refers to the model's performance on the **ImageNet** validation dataset. **Depth** refers to the topological depth of the network, this includes activation layers, batch normalization layers etc. **Time per inference** step is the average of 30 batches and 10 repetitions.*

3.5.1 working on benchmark models without using weights

working on one of the models that has proven its effectiveness in other areas of object recognition systems such as: "InceptionV3, DenseNet, VGG16, VGG19, ResNet50" without using weights, was our first option.

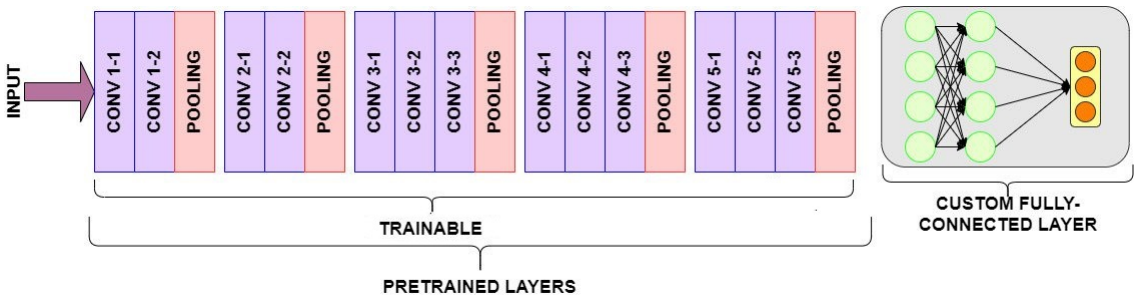


Figure 3.21: Benchmark Model Training process -VGG16 exemple-

To reuse these models (VGG16, VGG19, InceptionV3, DenseNet or ResNet) for COVID-19 classification, we get the dataset of chest X-ray images containing COVID-19 positive cases and normal cases. And applied the preprocessing steps. Then import the models without their pre-trained weights and fully connected layers. Add custom fully connected layers for COVID-19 classification on top of the models bases, we don't freeze any layer to retrain the whole model. We compile the model using Adam optimizer and a categorical cross-entropy loss. Train the model on the dataset, monitoring metrics such as accuracy, and evaluate its performance on a the test dataset using metrics like precision and recall. Finally, we used the model for inference on new chest X-ray images to classify COVID-19 cases and deploying it for real-time classification and calculating the memory and time coast. We tried to adjust hyperparameters based on dataset characteristics and requirements.

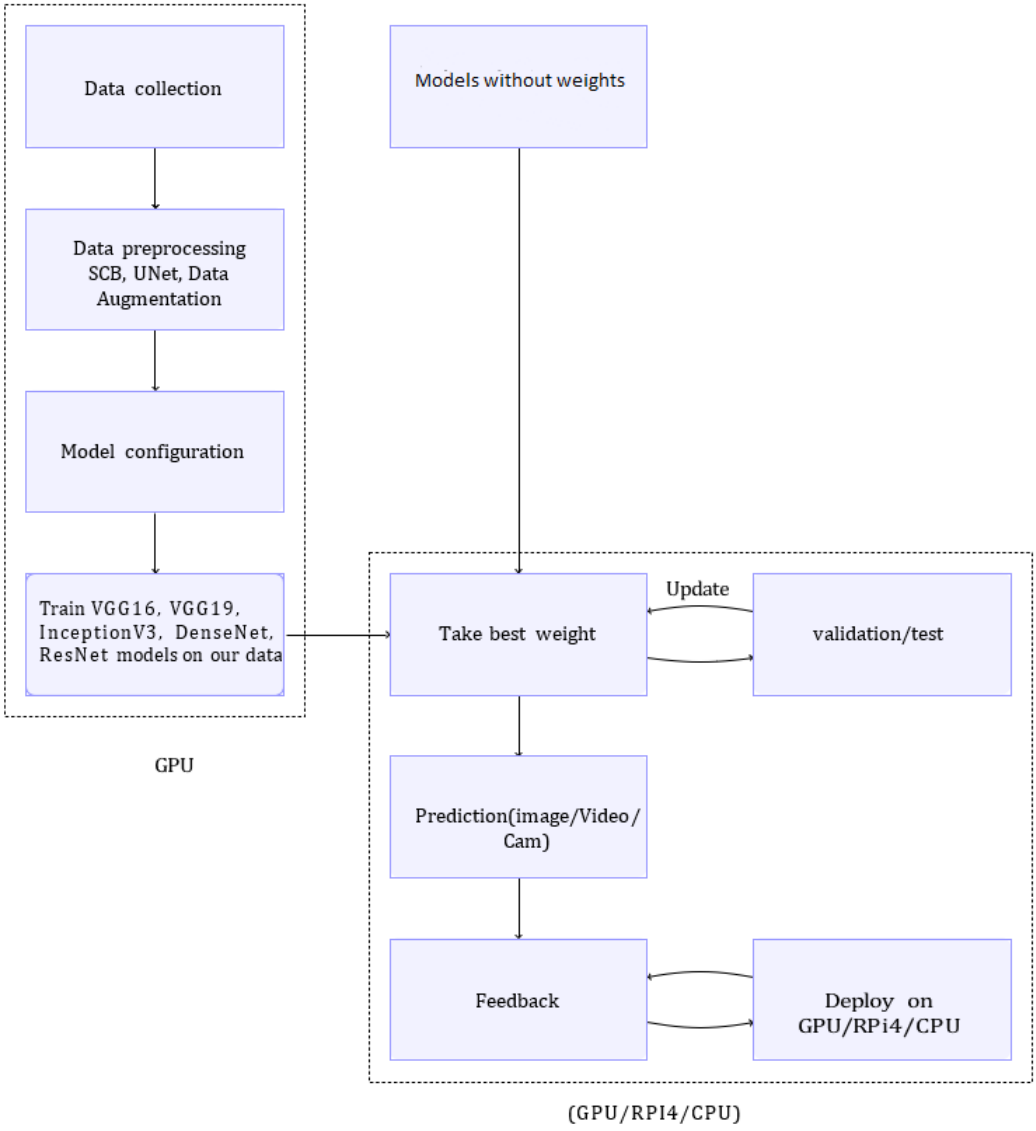


Figure 3.22: Benchmark Models training process -for all the models-

3.5.2 Transfer Learning using bottleneck features or Fine Tuning

There are two primary types of transfer learning:

0. **Transfer learning via bottleneck feature extraction:** We remove the FC layer head from the pre-trained network and replace it with a softmax classifier. This method is super simple as it allows us to treat the pre-trained CNN as a feature extractor and then pass those features through a Logistic Regression classifier.

0. **Transfer learning via fine-tuning:** When applying fine-tuning, we again remove the FC layer head from the pre-trained network, but this time we construct a brand new, freshly initialized FC layer head and place it on top of the original body of the network. The weights in the body of the CNN are frozen, and then we train the new layer head (typically with a very small learning rate). We may then choose to unfreeze the body of the network and train the entire network.

The first method tends to be easier to work with, as there is less code involved and fewer parameters to tune. However, the second method tends to be more accurate, leading to models that generalize better.

3.5.2.1 Using the bottleneck features of a pre-trained network

Since image recognition is a generic skill, and deep learning models are intrinsically very repurposable (reusable), So, an other refined approach would be to leverage a network pre-trained on a large dataset. Such a network would have already learned features that are useful for most computer vision problems, and leveraging such features would allow us to reach a better accuracy than any method that would only rely on the available data.

For example, you can take, an image classification model pre-trained (network + weights) on a large-scale dataset where it might be easier to acquire millions of images with relative ease, then apply transfer of learned skills with small modifications (remove the Fully Connected layers, also called top layers or Dense layers) to a significantly different problem (medical images); Doing so, it will remove the pre-trained weights at those layers.

Now the original network before the Fully Connected network is frozen so that, when the training is started on the new data set, only the newly added Fully Connected network are trained. Here the input to the FC network is called the bottleneck features. They represent the activation map from the last convolution layer in the network, by replacing and modifying the last layer of the model as needed and re-training in what is known as transfer learning, that *is usually done for tasks where your dataset has too little data to train a full-scale model from scratch.*

Hence with frozen weights in the main network, we will get to use the pre-trained weights to get important feature activations as bottleneck features into our newly added FC network, and now this new FC network (trained on our data) gives us the required wights.

Many pre-trained models (usually trained on the ImageNet dataset) are now publicly available for download and can be used to generate powerful vision models out of very little data, so we attempted this option with some of the well-known pre-trained models on the ImageNet dataset (VGG16, VGG19, Inception V3, DenseNet and ResNet50). The biggest problem is the ImageNet dataset doesn't contain x-ray or any medical images in the 1000 classes, those models will find difficulties to fine tune the weights or the features they learned from imageNET to use them on our classification problem and our little dataset.

The most common incarnation of transfer learning by using the bottleneck features is the following workflow:

1. Instantiate and Take layers from a base model without the top layers and load pre-trained weights into it.

2. Freeze all the first layers, to avoid updating/destroying any of the information they contain during the re-training.

3. Add new, trainable layers on top of the frozen layers. So they learn to turn the old features/ weights into predictions on a new dataset.

4. Train the new layers on our dataset.

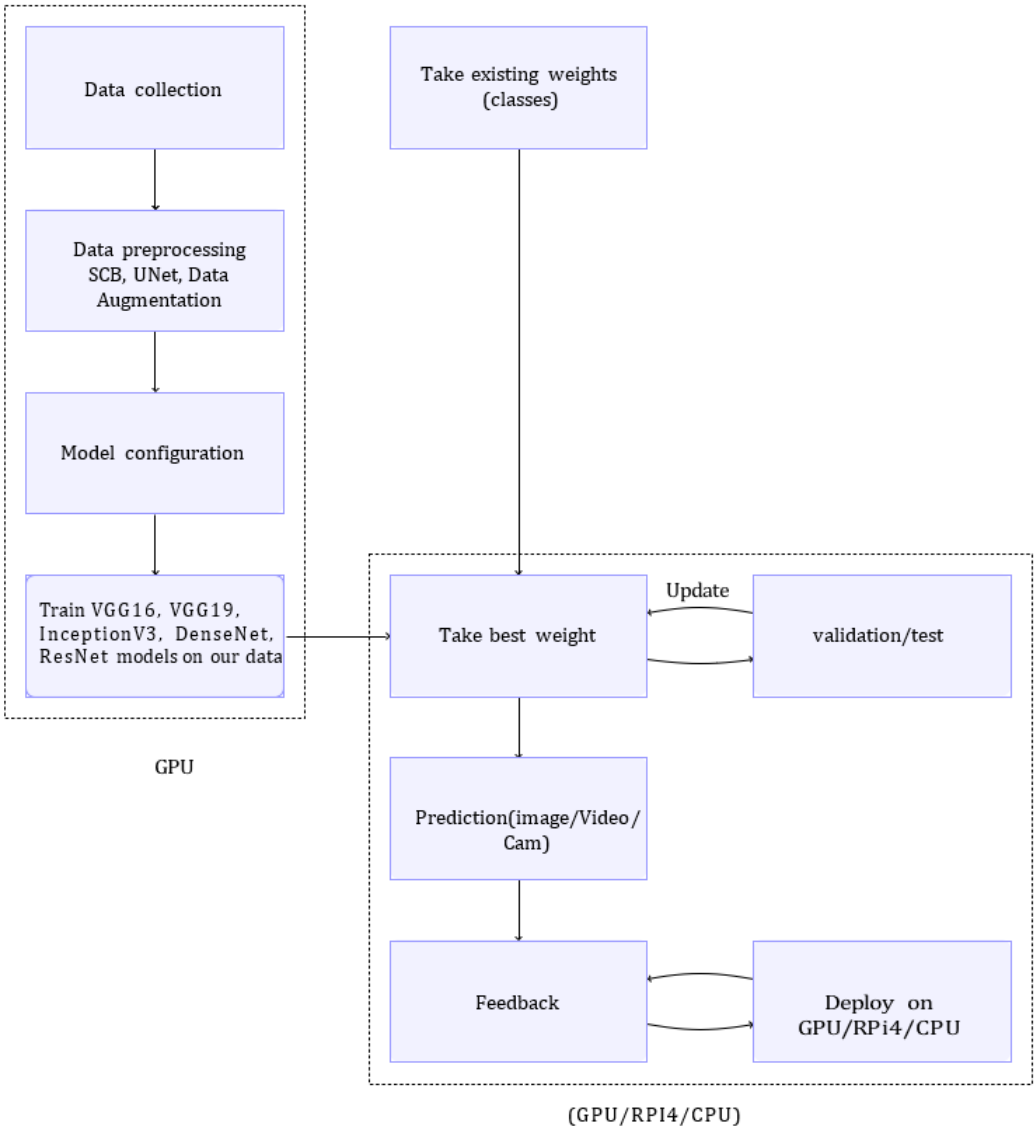


Figure 3.23: bottleneck features process -for all the models-

3.5.2.2 Fine-tuning the last layers of these pre-trained networks by freezing all the first layers

Fine-tuning refers to the process of further training a pre-trained neural network on a new dataset or task. When fine-tuning the last layers of a pre-trained network by freezing all the first layers, it means that the parameters (weights and biases) of the first layers are kept fixed and not updated during training, while only the parameters of the last layers are updated.

A key advantage of this workflow is that you only run the base model once on your data, rather than once per epoch of training. So it's a lot faster & cheaper.

But, the issue with that workflow, is that it doesn't allow you to dynamically modify the input data of your new model during training, which is required when doing data augmentation.

Here's a step-by-step explanation:

(a) **Pre-trained network**: Start with a neural network that has been pre-trained on a large dataset, typically for a general task like image classification. The pre-trained network already has learned features that are useful for many tasks.

(b) **Freeze first layers**: "Freezing" the first layers means that the parameters of these layers are not updated during the fine-tuning process. Since the lower layers of a neural network typically learn low-level features like edges and textures that are generally applicable across different tasks, keeping them fixed helps to retain these learned features.

(c) **Fine-tune last layers**: The last layers of the pre-trained network are typically fully connected layers or the output layer that are specific to the original task the network was trained on. In fine-tuning, these last layers are modified to adapt the network to the new task or dataset. This involves updating the parameters of these layers by training the network on the new dataset while keeping the parameters of the first layers fixed.

Fine-tuning the last layers while freezing the first layers allows the model to learn task-specific features from the new dataset while leveraging the learned features from the pre-trained network.

This approach is commonly used in transfer learning, where knowledge gained from one task or dataset is transferred to a new related task or dataset to improve performance, especially when the new dataset is small or similar to the original dataset.

in general, such network would have already learned features that are useful for most computer vision problems, and leveraging such features would allow us to reach a better accuracy than any method that would only rely on the available data. But because the ImageNet dataset doesn't contain x-ray or any medical images in the 1000 classes, those models will encounter difficulties to fine-tune the weights or the features they learned from ImageNet to use them on our classification problem and our little dataset.

The solution for that is to do a second round of Fine Tuning.

Once your model has converged on the new data, you can try to unfreeze all or part of the base model and retrain the whole model end-to-end with a very low learning rate.

This is an optional last step that can potentially give you incremental improvements. It could also potentially lead to quick overfitting.

It is critical to only do this step after the model with frozen layers has been trained to convergence. If you mix randomly initialized trainable layers with trainable layers that hold pre-trained features, the randomly-initialized layers will cause very large gradient updates during training, which will destroy your pre-trained features.

It's also critical to use a very low learning rate at this stage, because you are training a much larger model than in the first round of training, on a dataset that is typically very small. As a result, you are at risk of overfitting very quickly if you apply large weight updates. Here, you only want to readapt the pre-trained weights in an incremental way.

Note that:

- in order to perform fine-tuning, all layers should start with properly trained weights: for instance, you should not slap a randomly initialized fully-connected network on top of a pre-trained convolutional base. This is because the large gradient updates triggered by the randomly initialized weights would wreck the learned weights in the convolutional base. In our case, this is why we first train the top-level classifier and only then start fine-tuning convolutional weights alongside it.

- we choose to only fine-tune the last convolutional block rather than the entire network in order to prevent overfitting since the entire network would have a very large entropic capacity and thus a strong tendency to overfit. The features learned by low-level convolutional blocks are more general, less abstract than those found higher-up, so it is sensible to keep the first few blocks fixed (more general features) and only fine-tune the last one (more specialized features).

After instantiating VGG16, VGG19, InceptionV3, DensNet or ResNet50 and loading their weights, we add our previously trained fully-connected classifier on top:

We then proceed to freeze all convolutional layers up to the last convolutional block:

Finally, we start training the whole thing, with a very slow learning rate

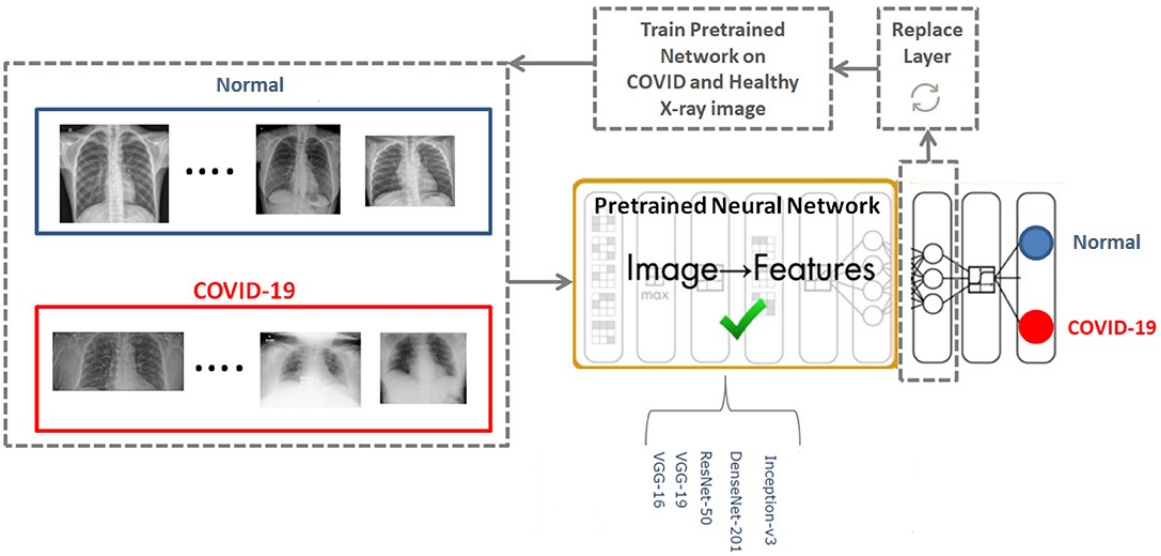


Figure 3.24: Benchmark Models Fine Tuning process -for all the models-

Fine-tuning can be summarized in the following multi-step process:

1. Remove the fully connected nodes at the end of the network (i.e., where the actual class label predictions are made).

2. Replace the fully connected nodes with freshly initialized ones.

3. Freeze earlier CONV layers earlier in the network (ensuring that any previous robust features learned by the CNN are not destroyed).

4. Start training, but only train the FC layer heads.

5. Optionally unfreeze some/all of the CONV layers in the network and perform a second pass of training.

3.5.3 training a small network from scratch (as a light weight baseline model)

the development of a personal model with the use of the largest number of parameters to reach the greatest effectiveness with the least number of examples from each class, in other word: "training a small network from scratch (**as a light weight baseline model**)" s our last option.

Despite numerous attempts to correct the situation of the previous options by changing the coefficient values to match the medical images, the results after more than 70 experiments (several cases with different parameters) were still overfitting. We noted that not all modifications produced the desired results due to the model's internal architecture, which is very deep or large in comparison to the number and quality of the data.

To deal with overfitting, the model's entropy capacity must be considered. A model that can store a large amount of data by utilizing more features has the potential to be more accurate, but it risks storing irrelevant features. A model that can only store a few features, on the other hand, will need to focus on the most significant features found in the data, which are more likely to be truly relevant and generalize better. As a result, we chose a simple CNN model (Figure 3.27) with a few layers, a few filters per layer, and many hyperparameters (learning rate, freeze, weight, last layer change, and early stop) in this paper to increase efficiency, avoid overfitting, reduce lose rate, and avoid the problems of previously tested models.

Due to the problems of other models which do not give the desired results in certain situations because their deep and wide internal structure which does not match the type of the data, especially in the field of chest medical images, so the optimal solution after experimenting several cases of different parameters is to create a model corresponding to the necessary depth and width conditions, as well as the appropriate number of layers, taking into account image size to maintain data quality and to train this model on the final dataset.

Now we go to principal step in our image classification system which is the CNN creation, and try to train it on our data. Since we only have few samples, our number one concern should be **overfitting**. Overfitting happens when a model exposed to too few samples learns patterns that do not generalize to new data, i.e. when the model starts using irrelevant features for making predictions.

This section we will describe how the convolutional neural network model was built and learned to detect COVID-19 in chest x-rays with only a few training samples per class. Because of the small amount of data, the classification problem is a difficult task in our case. Indeed, a small-scale collected dataset can be costly or even impossible to obtain, particularly in the field of medical imaging during a pandemic. Given the limited number of samples to learn and store, the goal is to find the best model that fits this type of information (X-rays), improve

accuracy, and minimize the loss function without using custom weights or feature engineering.

Meanwhile, a model that can only store a few features will have to focus on the most significant features found in the data, and these are more likely to be truly relevant and to generalize better, (we picked only the color-balanced region of interest “lung region only”), "and that’s why we have chosen to use a medium model rather than using a very deep one" and helping him by the previous preprocessing steps, that’s why our choice to modulate entropic capacity is by optimizing the number of layers and the size of each layer.

In our case, we will use a very small CNN with few layers and few filters per layer, *alongside* ***data augmentation*** (so our model would never see twice the exact same picture. This helps prevent overfitting and helps the model generalize better). **The more training samples you introduce into the network and the better they represent the variance of the real population, the better the network performance** and early stopping (by monitoring while training the error on the validation set after each iteration, and stopping the training when the network starts to overfit the data). (both early stopping and data augmentation tend to disrupt random correlations occurring in our data).

Therefore, the structure of the model can be determined by selecting the hyper parameters of the model, which are:

- The number of convolutional layers.
- The type of activation functions for each layer.
- The number of hidden units per layer.

In this model, which contains many layers, and through much experimentation, the result of choosing these hyper-parameters was the best among many other options.
To explain our model, it’s a simple stack of convolution layers with a ReLU activation and followed by max-pooling layers. On top of it we have two fully-connected layers. **We end the model with a two unit instead of one to give a percentage decision value better then binary decision for the prediction, and a SoftMax activation, which is perfect for a multi-classification**. To go with it we will also use the categorical-cross entropy loss function and adam optimizer to train our model.
The model is built to meet the required depth and width conditions, as well as the appropriate number of layers. To maintain data quality and train this model with the final database, image size is considered. The following hyper-parameters are used by the proposed CNN:

- Five layers of convolution2D that differ in the number of filters, so that it doubles as you go deeper, starting with the first layer, which contains 32 filters, and ending with the last layer, which contains 512 filters, all in order to extract many features.
- Five layers of max pooling in order to extract important information from the previous convolution2D layer.
- Flattening layer to render the information in one dimension.

- Then a Dense layer with 128 units.
- A Dense layer with two units based on the number of final classes was used, and we treated the problem as a multiclass problem instead of a binary classification problem. In addition, we used the SoftMax function to replace the classical sigmoid function usually used in this kind of case, since the Sigmoid gives as output a single number that represents the probability of belonging to a class. While, SoftMax is vectorized, which means that the model takes a vector with two classes and outputs another vector with probabilities of belonging to every class, and we *also used a categorical_crossentropy loss function and adam optimizer to train our model.*
- In all these layers, the ReLU activation function was used because it is the most advanced compared to the other functions.

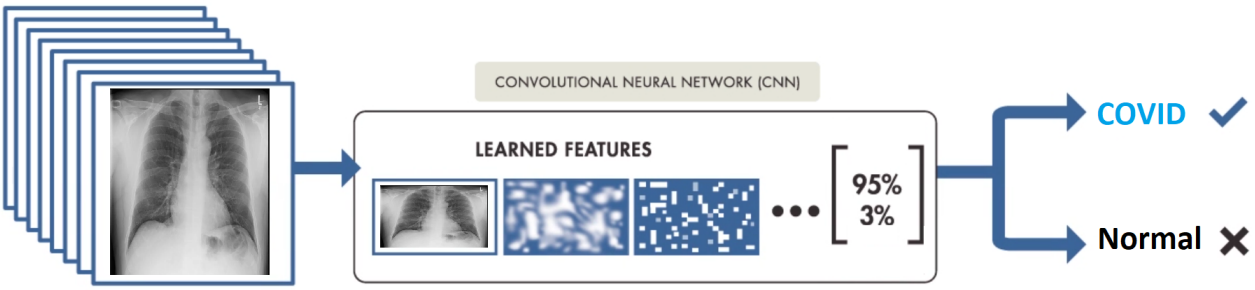


Figure 3.25: CNN Training From Scratch

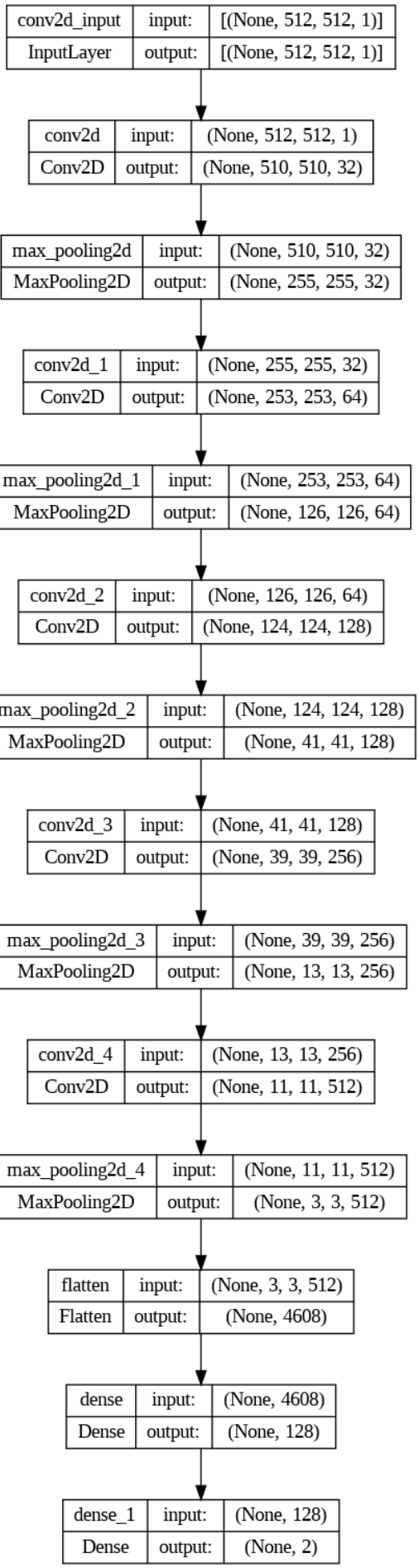


Figure 3.26: CNN network Architecture

The performance of our proposed approach and model was compared with those obtained by applying transfer learning and fine tunig some of the most widely used convolutional neural network models (with different type of architectures "VGG16, VGG19, ResNet, Inception V3, and DenseNet" and different hyperparameters "freezing/unfreezing layers, using pretrainned weights or Not, Learning Rate, early stopping.."). The results showed that the proposed model preprocessing steps of color balance and lung segmentation combined with lightweight CNN with his low number of computational layers outperforms the pre-trained benchmark models, achieving the best accuracy, and more then that it was the best in generalization when confronted to five external datasets.

So, the proposed lightweight model achieved the best overall result in classifying lung diseases allowing it to be used on devices with limited computational power. On the other hand, all the models was validate on the test dataset showed good metrics but a poor accuracy and hight loss in real world scenario when confronted on data they never saw before.

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 510, 510, 32)	320
max_pooling2d_5 (MaxPooling 2D)	(None, 255, 255, 32)	0
conv2d_6 (Conv2D)	(None, 253, 253, 64)	18496
max_pooling2d_6 (MaxPooling 2D)	(None, 126, 126, 64)	0
conv2d_7 (Conv2D)	(None, 124, 124, 128)	73856
max_pooling2d_7 (MaxPooling 2D)	(None, 41, 41, 128)	0
conv2d_8 (Conv2D)	(None, 39, 39, 256)	295168
max_pooling2d_8 (MaxPooling 2D)	(None, 13, 13, 256)	0
conv2d_9 (Conv2D)	(None, 11, 11, 512)	1180160
max_pooling2d_9 (MaxPooling 2D)	(None, 3, 3, 512)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_2 (Dense)	(None, 128)	589952
dense_3 (Dense)	(None, 2)	258

Total params: 2,158,210
Trainable params: 2,158,210
Non-trainable params: 0

Figure 3.27: CNN network Outputs and parameters of every layer

3.5.3.1 Mathematical Reasoning of the proposed CNN

In this section, a mathematical reasoning of the proposed CNN is provided.

Convolution The convolution operation in CNNs involves applying a filter (kernel) to an input image to extract features. Let X be the input image, W be the filter, and C be the resulting feature map. The convolution of the input image X with filter W at a specific location (i, j) in the feature map is given by:

$$C(i,j) = \Sigma_{m=0}^{M-1}\Sigma_{n=0}^{N-1} X(i+m, j+n) \cdot W(m,n) \tag{3.5}$$

Where $C(i, j)$ is the value of the feature map at position (i, j). M and N are the dimensions of the filter W. $X(i + m, j + n)$ represents the pixel values of the input image at location $(i + m, j + n)$. $W(m, n)$ represents the filter weights at position (m, n).

Activation functions Activation functions introduce non-linearity into the CNN. One of the commonly used activation functions is the Rectified Linear Unit (ReLU):

$$ReLU(x) = max(0, x) \tag{3.6}$$

where x is the input to the ReLU function.

Pooling Max-pooling is a popular pooling technique used to down-sample the feature maps and reduce spatial dimensions. It selects the maximum value from a region of the feature map. Let P be the pooled output, and S be the size of the pooling window.

$$P(i,j) = max_{m=0}^{S-1} max_{n=0}^{S-1}\ C(S \cdot i + m, S \cdot j + n) \tag{3.7}$$

where $P(i, j)$ is the pooled value at position (i, j). $C(S \cdot i + m, S \cdot j + n)$ represents the values in the feature map that fall within the pooling window.

Loss functions For classification tasks, the Cross-Entropy Loss function is commonly used. Suppose we have N training samples, and y_i represents the ground truth label for the $i - th$ sample, and p_i represents the predicted probability for the $i - th$ sample. The Cross-Entropy Loss is given by:

$$CE = -\frac{1}{N}\Sigma_{i=1}^{N}\Sigma_{c=1}^{C} y_i^c \cdot log(p_i^c) \tag{3.8}$$

where CE is the Cross-Entropy Loss. C is the number of classes. y_i^c is the indicator function that takes the value 1 if the true label for the $i - th$ sample is c, and 0 otherwise. p_i^c is the predicted probability for the $i - th$ sample belonging to class c.

Softmax Softmax is a mathematical function used in classification tasks to convert raw scores or logits into probabilities. It takes a vector of arbitrary real-valued scores $z = (z_1, z_2, ..., z_k)$ as input and outputs a vector of probabilities $p = (p_1, p_2, ..., p_k)$, where p_i represents the probability of the input belonging to the i^{th} class.

The softmax function is defined as follows:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \tag{3.9}$$

for $i = 1, 2, ..., k$.

In this formula, e is the base of the natural logarithm (Euler's number), z_i is the raw score or logit for the i^{th} class, and $\sum_{j=1}^{k} e^{z_j}$ is the sum of the exponentiated raw scores across all classes.

The softmax function normalizes the raw scores such that the resulting probabilities sum up to 1, making it suitable for interpreting the output as class probabilities in a classification problem.

Optimizer Adam Adam (Adaptive Moment Estimation) is an optimization algorithm used for training deep learning models. It combines the advantages of two other popular optimization algorithms, namely AdaGrad and RMSProp, to adaptively adjust the learning rate for each parameter.

The Adam optimizer updates the parameters of the model using the following formulas:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3.10}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{3.11}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3.12}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{3.13}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{3.14}$$

where: - m_t and v_t are the first and second moment estimates of the gradients, respectively, - g_t is the gradient of the parameters at time step t, - $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected estimates of m_t and v_t, - β_1 and β_2 are exponential decay rates for the moment estimates, - η is the learning rate, - ϵ is a small constant to prevent division by zero.

Adam adjusts the learning rate for each parameter based on the magnitude of the gradients and their past history, allowing it to converge faster and potentially avoid local minima.

3.5.4 Training and Tuning

Training is an important step in developing a powerful model. The process in this study begins with an unknown number of epochs and employs an early stopping function to avoid overfitting, while validation monitors the accuracy rate and error rate. It is critical to remember that model training begins from the beginning. The model is evaluated using data that has never been used for training. It demonstrates how the model might behave in the real world when confronted with data it has not yet seen. Because we started with parameters that were assumed to be implicit, the next step is to tune the parameters to improve the model's learning. Many values

were tested, considering epochs and batches. The learning rate is another parameter that defines how far the line is moved at each step based on the previous step. In other words, the learning rate governs our neural network's weights in relation to the loss gradient. These values have a significant impact on the model's accuracy and learning time. The compile method was used to create the training model, which has the following parameters:

- **Loss function:** we use the Categorical Cross entropy loss function since there are two or more classes of labels.
- **Type of optimization:** The optimizer determines the optimization algorithm used during training. The most commonly used optimizer is Adam, which combines the advantages of Adagrad and RMSprop. the Adam optimizer is used because it is the fastest method, and it quickly converges when correcting the latency of the learning rate and the high contrast.
- **Metrics:** the accuracy metrics are used to evaluate the performance of the model. This metric creates two local variables, total and count, which are used to calculate the frequency at which the predicted class matches with the correct class. This frequency is finally returned as a precision that simply divides the total by the count.
- ***Target_size = (512, 512):*** The image size affects the model's performance and training time. A larger image size can improve the model's accuracy but will require more memory and computational power. We use an image size of 512x512.
- ***Batch_size = 8 :*** The batch size is the number of samples processed in one training iteration. It affects the memory usage and training time. A larger batch size may require more memory, but it can also speed up the training process. this batch is selected to work with less effort and memory and to have a moderate fluctuation of the training since the database is not very large.
- ***Class_mode = 'categorical' :*** is used to label every class.
- **Learning rate:** The learning rate determines the step size taken during the optimization process. A high learning rate can cause the model to overshoot the optimal solution, while a low learning rate may result in slow convergence. We use a learning rate of 0.0001 and 0.000001.
- **Number of epochs:** The number of epochs is the number of times the entire dataset is passed through the model during training. we used two options fixed number of epochs and early stopping.

In addition, for better training and to avoid overfitting, an Early Stopping regularization technique is used as gradient descent. In each iteration, this method updates the model to better fit the training data and improves its performance on data outside the training set. This technique's parameters are as follows:

- a monitor with the '*val_loss*' parameter,
- patience equals 7,
- automatic mode.

3.5.4.1 The number of epochs

Forthe number of epochs: We opted for some options for the training, and the choice of the iterations and epochs was one of them. So we tried to do so, with an open number of epochs with an early stopping option to avoid that our model overfit, and we tried also to set up empirically the number of epochs (we tried from 30 epochs up to 120 epochs values).
To choose the best number of epochs to train our model (CNN) we took various factors, including the complexity of the dataset, the size of the dataset, the architecture of the CNN, and our specific addressed problem.
We experimented with different numbers of epochs and monitor the model's performance on test sets to determine the best number to evaluate the model's stability and generalization performance across different datasets.
Even though, there is no universally optimal number of epochs that applies to all cases.
We tried two options (fixing the number or epochs and using early stopping):

- First option: fixing the number of epochs while considering those guidelines when determining the number of epochs: (Monitoring Validation Loss, Generalization Performance, Complexity of the Dataset and Model (If large and complex ⇒ more epochs to allow the model to learn intricate patterns and achieve good performance. If relatively small or the model is simple, ⇒ it may converge quickly, requiring fewer epochs), Hyperparameter Tuning).

- *Second option: by using EarlyStopping*
  Early stopping in CNN training is motivated by preventing overfitting, saving computational resources, determining automatically the optimal training epoch, improving generalization performance, and handling limited data effectively.

3.5.4.2 The Learnin Rate

We opted for to options of learning rate: No learning rate and the second using a the learning rate. the "No learning rate" term doesn't mean a "zero" learning rate, but it means we took the default little value of 0.001 learning rate...
And when we use the option of learning rate that means we use an extremely small value of 0.00001, that makes the updates to the model parameters very minute and incremental)
The learning rate, denoted by the symbol α, is a hyper-parameter used to govern the pace at which an a machine learning model updates or learns the values of its parameters during the training process. In other words, the learning rate regulates the weights of our neural network concerning the loss gradient, in response to the estimated loss error or stochastic gradient descent optimization algorithm.
The learning rate is one of the most important hyperparameters when we started configuring our neural network. The choice of learning rate can have a significant impact on the model's convergence and performance. Choosing the suitable learning rate was challenging as a value

too small may result in a *long training process* that could get stuck, while a value *too large* may cause *instability or prevent convergence*, and result in learning a sub-optimal set of weights too fast or an unstable training process.
Experimentation was the most important to determine the optimal learning rate for a specific problem.
when we plots accuracy graph it shows oscillations in behavior for the too-large learning rate of 1.0 and the inability of the model to learn anything with the too-small learning rates of 1E-7. We have seen that the model was able to learn the problem well with the learning rates 1E-3 and 1E-5 (1E-3 = we called it "No learning rate" because it's used by defaults in most of the framworks like tensorflow and keras). (1E-5 = we called "learning rate" which is a more lower rate), although successively slower as the learning rate was decreased. In summary, the main differences between the learning rate values lie in the speed of convergence and the risk of overshooting or getting stuck in local optima. Higher learning rates (e.g., 1.0) can lead to faster convergence but may be prone to divergence. Smaller learning rates (e.g., 0.001, 0.000001) offer more stability but require longer training times. after many experimentation and tuning, and depending on the specific problem, the dataset, and the characteristics of the model. The results show that a learning rate of 0.00001 results in good model performance on the train and test sets.

Table 3.7 shows the hyperparameter configuration and nomenclature including all variables and parameters with the used values.

Hyperparameter	Value
Data augmentation	normalization, rotations, scale changes (zoom in/out), horizontal flipping
SCB parentage	0.01 or 1%
UNet Bounding boxes	(1) The two largest connected regions with eight connectivity pixels. (2) a small safety distance of 5 pixels to each side was added, to the initial bounding box at the top, left, right and bottom of the bounding box
UNet Batch size	4
UNet Epochs	50 epochs on average (but stopped on 40 to avoid overfitting
UNet last layer	final convolutional layer with a softmax activation function was applied to model output with Adam optimization criterion and 0.0005 learning rate
Image size (IN/OUT)	512*512
CNN Learning rate	0.0001 for the NoLR or 0.000001 for the LR
CNN Batch size	8
CNN Number of training epochs	(1) 120 and early stopping (which stops always between 25-30) for the proposed model. (2) 30 for the pre-trained models, (3) When early stopping is used : (monitor) = 'val_loss' parameter, patience = 7, mode = automatic)
CNN optimizer	Adam
CNN loss Function	Categorical Cross Entropy

Tableau 3.7: Nomenclature of variables

training procedure:
after the configuration of the model we start the training.

(1) Feed the preprocessed images to the model.

(2) Calculate the total loss using the defined loss function.

(3) Perform backpropagation to compute the gradients.

(4) Update the model's weights using the chosen optimizer, and save the best weights.

(5) Repeat the above steps for a specified number of epochs.

(6) Evaluate the trained model on a separate test set to assess its performance.

3.6 Experiments

3.6.1 Settings

The experiments were carried out in the Google Colab environment with Python and a GPU for learning. The dataset was divided into 25% for learning and 75% for testing. TensorFlow, OpenCV, and Scikit-learn were among the Python packages used to preprocess the dataset, train the CNN model, and evaluate its performance. The model's performance was assessed using metrics such as TP, FP, TN, FN, accuracy, precision, recall, and F1 score, as well as the Receiver Operating Characteristic (ROC) curve, confusion matrix, the loss and accuracy graph were monitored while training was processed in every epoch. Several hyperparameters, most notably learning rate, weights, and freeze/unfreeze layers, guided the training process. These hyperparameters were carefully chosen and tuned to improve model performance. To ensure optimal performance, an early stop was used after training with 120 epochs to monitor the progress of the training process and avoid overfitting.

3.6.2 Performance evaluation

The proposed metrics to evaluate the proposed work, to make visible specific quantitative measures or indicators that can be used to assess the performance, effectiveness, and impact of our research. Metrics provided objective criteria for evaluating the outcomes and results.
we can say that most of the formulas for calculating Precision, Recall, F1-Score, Accuracy, Loss, and ROC (Receiver Operating Characteristic) are using TP, FP, TN, FN, here are some definitions of the most essential terms:

Confusion Matrix: Confusion matrix are tables that describe how well a model performs on test data and allow for the calculation of actual values. Instead of rates.
True positives (TP): Cases where we predicted yes (the patient has the disease) and, in fact, they have the disease.
True negatives (TN): We expected no, and they aren't infected.
False positives (FP): Yes, as expected, but they don't actually have the virus.
False Negatives (FN): We had expected no, yet they do have the illness.

Precision: Precision measures the proportion of correctly identified positive instances out of all instances predicted as positive, which is an indicator of the model's success.

$$Precision = (TP)/(TP + FP) \tag{3.15}$$

Recall: Recall, also known as sensitivity or true positive rate, measures the proportion of correctly identified positive instances out of all actual positive instances.

$$Recall = (TP)/(TP + FN) \tag{3.16}$$

F1-Score: The F1-Score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance and for how accurate models are on a given dataset. It's used to assess binary classification systems that divide the world into positive and negative categories.

$$F1 - Score = 2 * (Precision * Recall)/(Precision + Recall) \tag{3.17}$$

Accuracy: Accuracy measures the proportion of correct predictions (both true positives and true negatives) out of the total number of instances; it assess the performance of classification and regression algorithms.

$$Accuracy = (TP + TN)/(TP + FP + TN + FN) \tag{3.18}$$

Loss: Loss is a measure of how well the model is performing. It quantifies the discrepancy between the predicted values and the true values. The choice of loss function depends on the specific problem and task, such as categorical cross-entropy for classification.

ROC (Receiver Operating Characteristic): ROC is a graphical plot that illustrates the performance of a binary classifier as the discrimination threshold varies. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.
To calculate the ROC, we need the True Positive Rate (TPR) and False Positive Rate (FPR). These can be calculated as follows:

$$TPR = (TP)/(TP + FN) \tag{3.19}$$

$$FPR = (FP)/(FP + TN) \tag{3.20}$$

These values can then be used to plot the ROC curve which quantifies the classifier's overall performance.
Also two typical CNN Curves are used and displayed for each model: Model accuracy curve, Model Loss curve).
All these metrics were calculated in both the training/test process step and the generalization on the external datasets step.
we also calculated the **time** (speed) in GPU and CPU, to prove that our approach can be used in embedded systems like RPi4 or other and in real-time since the FPS and the processing time is acceptable for an image of 512x512 resolution. and the pre-processing steps are low-consuming time compared to their benefits for the global system's accuracy.
So the system can be used by doctors with phones or raspberries in real time to detect, segment, and classify lungs infected with covid-19.

Time: was calculated from the mean inference time of every image from the test set.

FLOPs: We have added complexity and FLOPs and parameter size metrics to calculate the size, weight, and computational consumption (in the table 3.10) of our CNN and the used pre-processing steps compared to the other models.
And ours is still low consuming in time and resources.

Formula for Calculating FLOPs

In the provided formula:

- N: Batch size.
- K: Number of output channels (filters).
- C: Number of input channels.
- H_{out}, W_{out}: Spatial dimensions (height and width) of the output feature map.
- H_k, W_k: Spatial dimensions (height and width) of the convolutional kernel/filter.
- P: Number of operations per output pixel (typically 2 for multiplication and addition).

The formula to calculate FLOPs (Floating Point Operations) is:

$$\text{FLOPs} = (2 \times N \times K^2 \times C \times H_{out} \times W_{out} \times H_k \times W_k) \times P \tag{3.21}$$

This formula estimates the total number of floating-point operations performed in a single forward pass of a convolutional layer in a neural network, taking into account the number of operations involved in the convolution operation (multiplications and additions) for each output pixel across all elements of the batch. The FLOPs was calculated for all the our models, the benchamrks (VGG16, VGG19, InceptionV3, DenseNet, ResNet) and UNet.

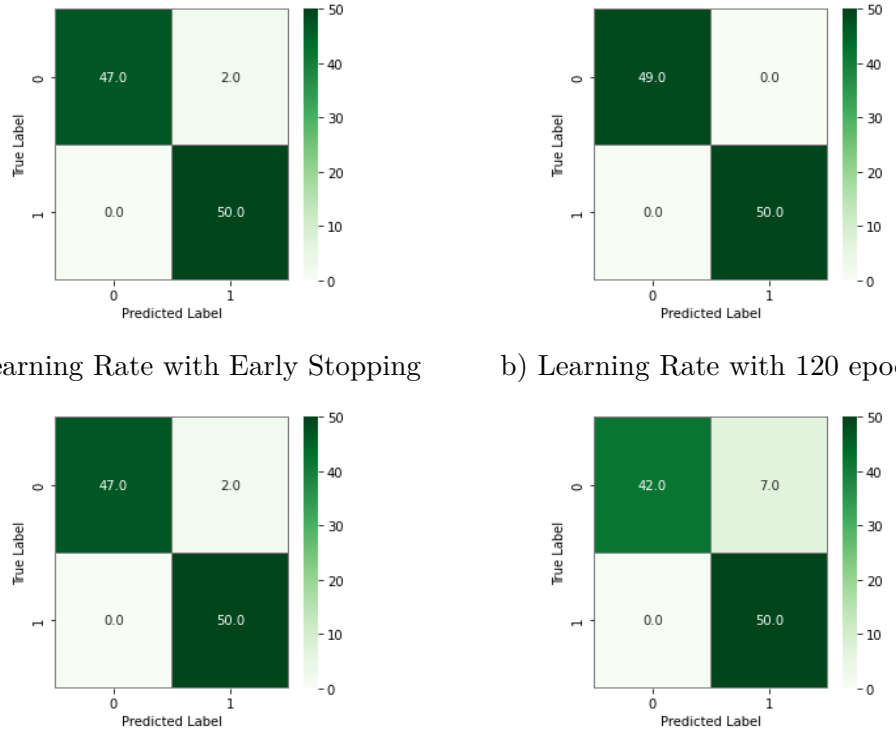
All these metrics accurately capture the important aspects of our research objectives and provide valuable insights, and provide a good quantitative comparison with the other models, and the compromise between the high accuracy and low time and computational consumption is optimal, and we outperform the other models in both accuracy and time, especially when it comes to generalize our model in external datasets.

3.6.3 Results and discussion

This section presents the results obtained from the test data, which contains 99 images. All the test images were subjected to the same preprocessing phase. Then, the model predicted the classes for these images and an estimation of its performance is made by comparing the number of correct predictions and errors. Figure 3.28 shows the confusion matrix, Figure 3.29 shows the accuracy and loss functions, Figure 3.30 shows the Receiver Operating Characteristic (ROC) curve, and Table 3.8 shows the quantitative results. Both results were obtained using four model configurations.

The results listed in Table 3.8 indicate that the model performs well with all options, particularly with a learning rate and 120 epochs, with a maximum efficiency of 100% in all

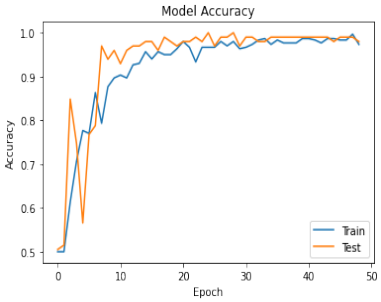
metrics. This is due to the proposed CNN's simplicity, which allows us to store only the significant features found in the data and thus generalize more effectively. This is only possible if the model makes use of the relevant data, which in our case comes from the color-balanced region of interest after the preprocessing step. Indeed, the model used in the cutting-edge takes the data as is, necessitating more layers and filters per layer, rendering data learning unnecessary.



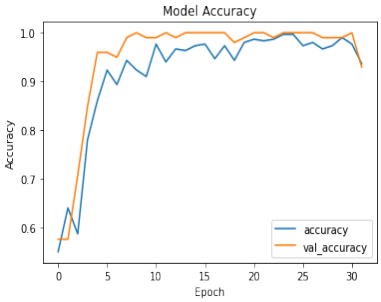
a) Learning Rate with Early Stopping b) Learning Rate with 120 epochs

c) No Learning Rate with 120 epochs d) No Learning Rate with Early stopping

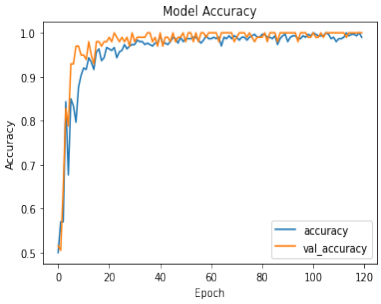
Figure 3.28: Confusion Matrix



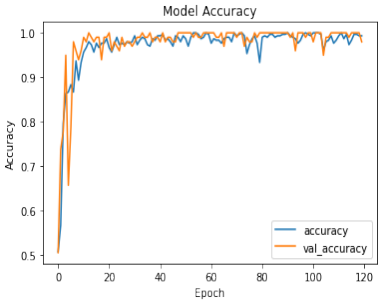
a) Learning Rate with Early stopping



b) No Learning Rate with Early stopping

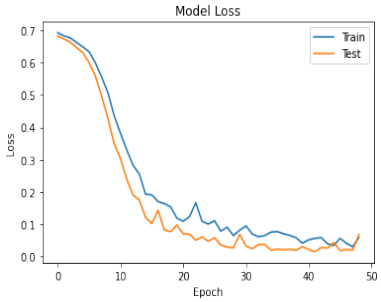


c) Learning Rate with 120 epochs,

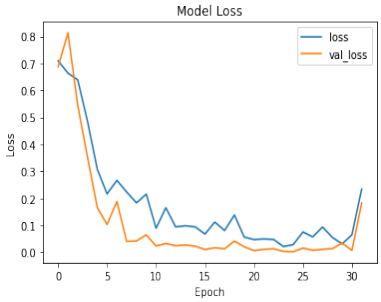


d) No Learning Rate with 120 epochs

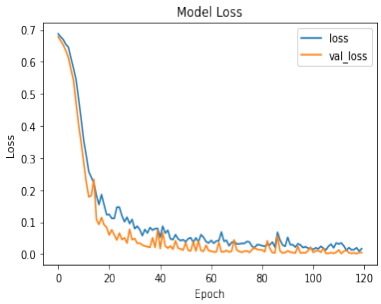
Accuracy



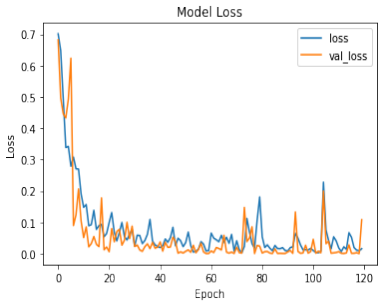
a) Learning Rate with Early stopping



b) No Learning Rate with Early stopping



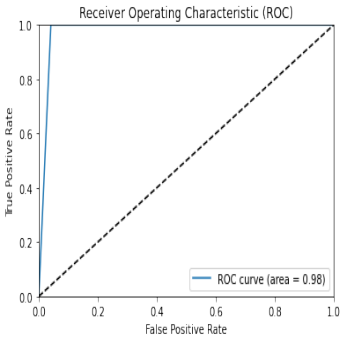
c) Learning Rate with 120 epochs,



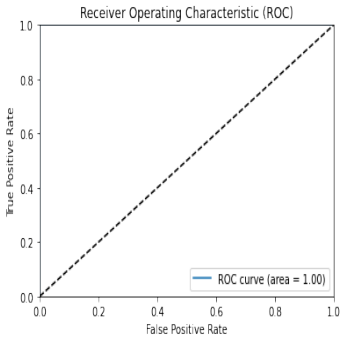
d) No Learning Rate with 120 epochs

Loss

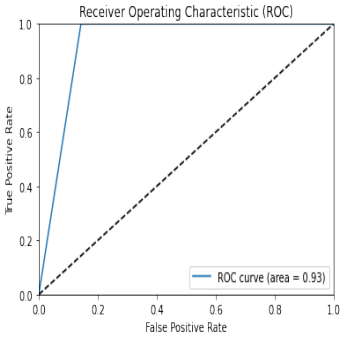
Figure 3.29: Accuracy and Loss



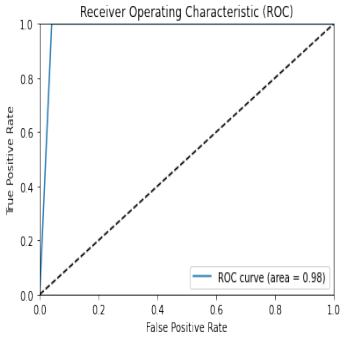
a) ROC Learning Rate with Early Stopping



b) ROC Learning Rate with 120 epochs



c) ROC No Learning Rate with Early Stopping



d) ROC No Learning Rate with 120 epochs

Figure 3.30: Receiver Operating Characteristic (ROC) curve

Tableau 3.8: The obtained results on well-known metrics with 4 model options.

Metrics		Precision		Recall		F1-Score		Acuuracy	Loss	ROC
Options		Covid	Normal	Covid	Normal	Covid	Normal			
LR	EarlyStop	1.00	0.96	0.96	1.00	0.98	0.98	0.98	0.0199	0.98
	120 epochs	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.0042	1.00
NoLR	EarlyStop	1.00	0.88	086	1.00	0.92	0.93	0.93	0.1392	0.93
	120 epochs	1.00	0.96	0.96	098	0.98	0.98	0.98	0.0066	0.98

In Table 3.9 we listed the best results of every model of the five deep learning methods. Precision, F1 score, accuracy, loss, and ROC metrics are calculated. Compared to Table 3.8 our proposed approach is outperforming them in many metrics, especially the most important ones (Accuracy and Loss).

Tableau 3.9: The results of well-known metrics with the highest scores of every best option of the five contestant models.

model	Precision	F1 score	accuracy	loss	ROC
InceptionV3_NOW_NOLR-freeze	86.20	92.59	91.92	0.220	0.92
VGG16_NOW_LR-NOfreeze	98.04	99.01	98.99	0.006	0.99
VGG19_NOW_LR-NOfreeze	98.04	99.01	98.99	0.005	0.99
VGG19_NOW_NOLR-freeze	92.59	96.15	95.96	0.319	0.96
ResNet_NOW_LR-freeze	90.91	95.24	94.95	0.171	0.95
ResNet_NOW_NOLR-freeze	96.15	98.04	97.98	0.014	0.98
ResNet_W_LR-freeze	89.28	94.34	93.94	0.184	0.94
DenseNet_NOW_NOLR-NOfreeze	98.04	99.01	98.99	0.008	0.99
DenseNet_NOW_LR-NOfreeze	90.91	95.24	94.95	0.417	0.95

In the following section, we present the results obtained from the test data, consisting of 99 images. All test images underwent the same preprocessing phase.

Subsequently, each model predicted the classes for these images, and we estimated their performance by comparing the number of correct predictions and errors. Figure 3.31 displays the accuracy graphs, while Figure 3.32 illustrates the loss graphs. All results were obtained using various configurations of transfer learning, bottleneck features, and the benchmark model with their original architecture.

To accomplish this, we experimented with several parameters including freezing layers, the learning rate, and the weights. For each option, we conducted experiments with and without these modifications. For instance, when we mention 'without weights,' it implies that we didn't load the pretrained weights of the model from the ImageNet dataset.

We explored eight options for every model among the five selected benchmarks (DenseNet, ResNet, InceptionV3, VGG16, VGG19). Consequently, we had a total of forty (40) trained model, from which we retained only those that yielded satisfactory results. the name of every corresponding model to its graph are in the to of every graph.

We did not include the other metric graphs of every model due to the large number of graphs. Consequently, we only retained the relevant ones in Table 3.9. This table includes metrics for models that exhibited good generalization results.

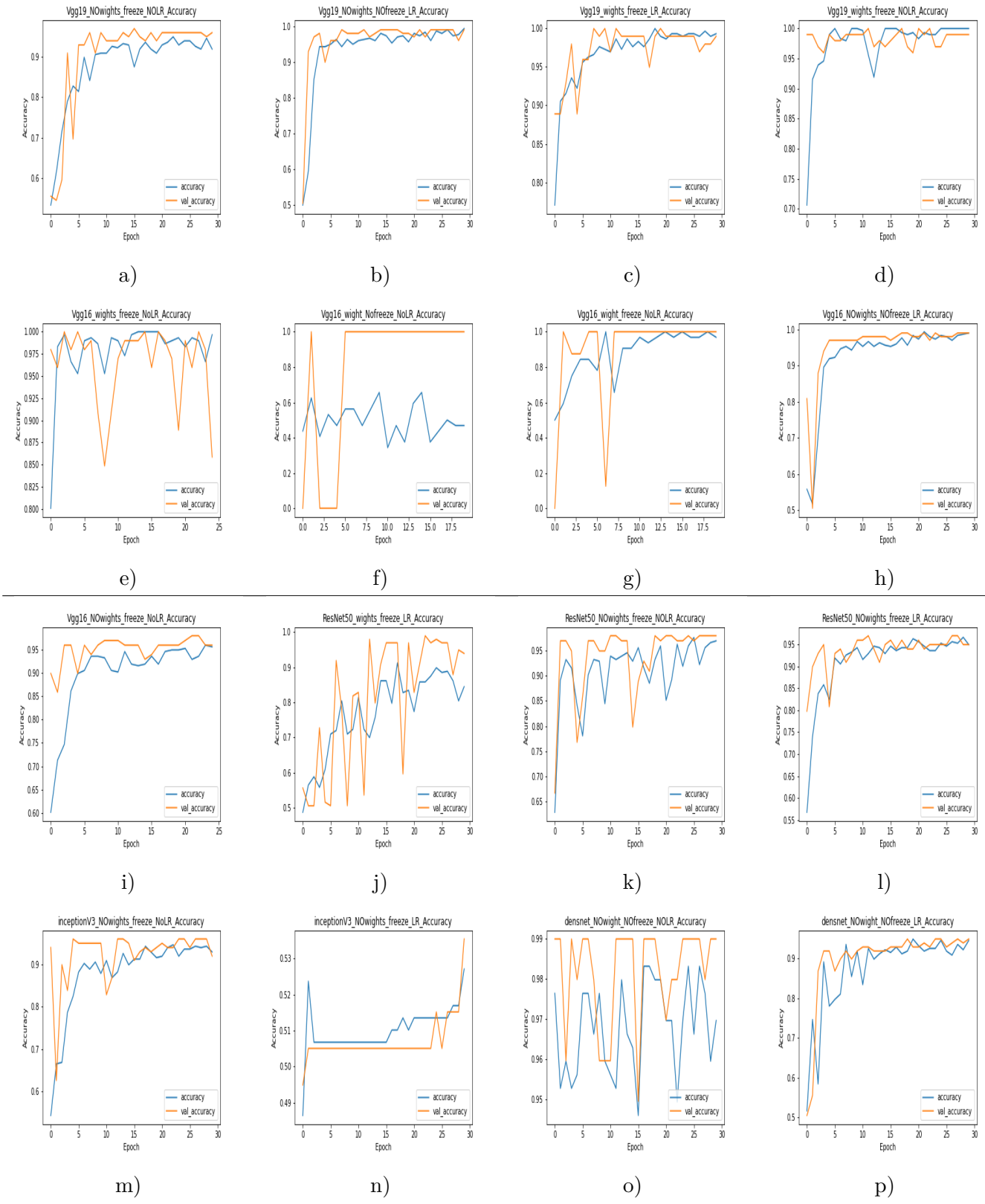


Figure 3.31: Best models accuracy on test set

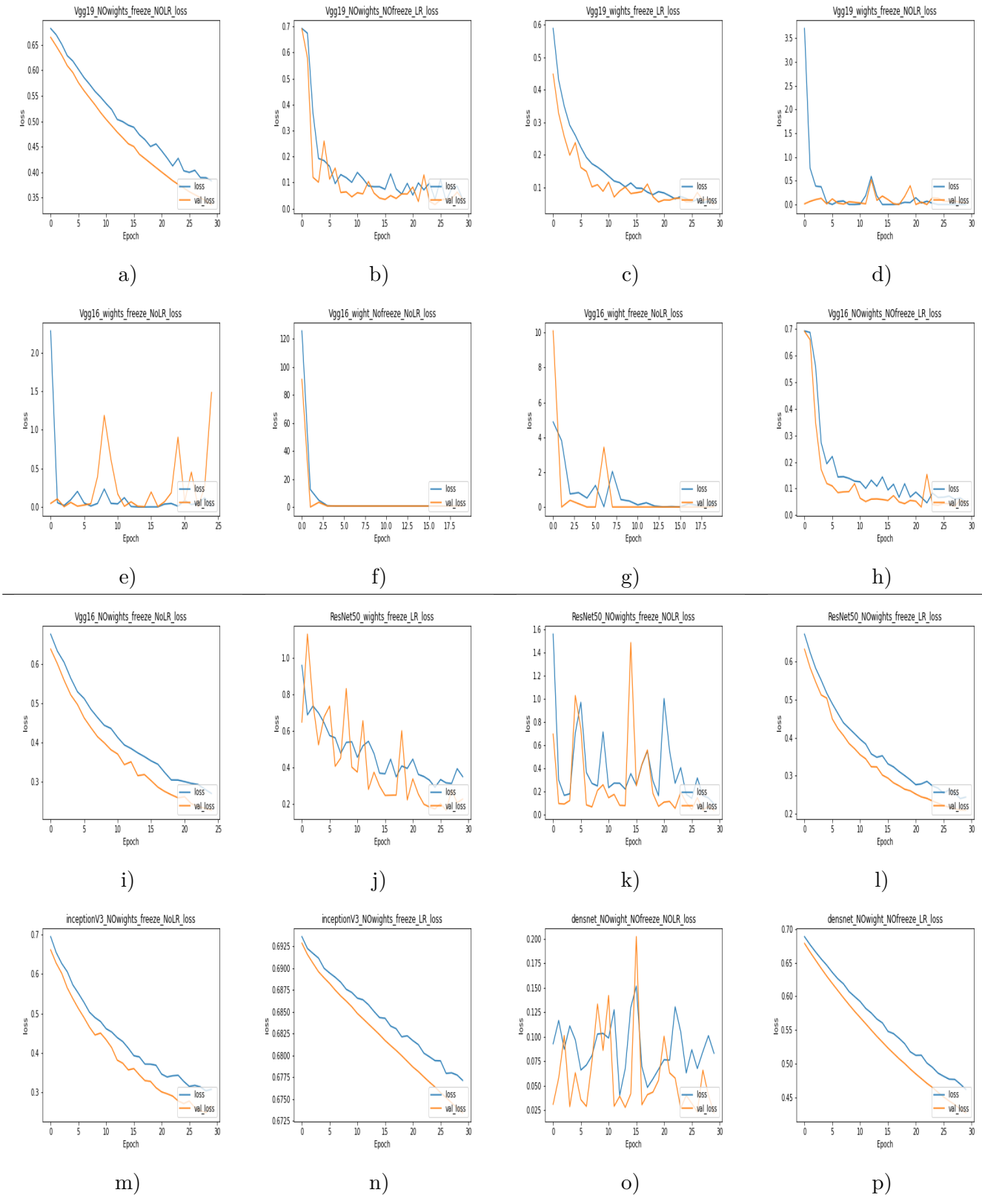


Figure 3.32: Best models Loss on test set

3.6.4 discussion with the advantage of the proposed method

For comparison with the proposed model, transfer learning models were used, namely, InceptionV3, ResNet50, DenseNet, VGG16 and VGG19, and other hyperparameters for fine-tuning,

and bottleneck. Transfer learning is a machine learning technique that depends on using the weights of pre-trained models as starting point or used from scratch for training the model on a new task using a new dataset, we added some fine-tuning by freeze/unfreeze layers except for the output layer units, besides changing the learning rate between medium and small parameters. Subsequently, the images in our dataset were fed into different pre-trained CNN models which have different input image sizes, number of layers, and number of parameters. In addition, softmax was used as an activation function in the output layer.

FLOPs and Complexity: Table 3.10 shows the dimensions of the input image, number of total parameters, number of trainable parameters, and the FLOPs for each of the neural network models (VGG16, VGG19, ResNet50, Inception V3, and DenseNet, and our lightweight CNN and the used Unet).

To calculate the FLOPs for each of the neural network models, we'll need to compute the number of operations involved in a forward pass through the model, such as convolution, pooling, and fully connected layers. The number of FLOPs gives you an estimate of the computational complexity of the model, which is useful for understanding its efficiency and performance. Each operation has a specific number of FLOPs associated with it.

Tableau 3.10: Number of parameters and trainable parameters and FLOPs in transfer learning models and the proposed model

Model Name	Input Image Dimensions	Total N° of parameters	trainable parameters	FLOPs per inference
VGG16	5125121	138,344,130	134,260,994	15.5 billion
VGG19	5125121	143,667,842	143,667,842	20 billion
DenseNet	5125121	15,000,000	15,000,000	15.3 billion
InceptionV3	5125121	30,000,000	30,000,000	6.6 billion
ResNet	5125121	26,000,000	26,000,000	4.1 billion
The proposed model	5125121 (UNet)	7,759,521	7,759,521	107,697,152
	5125121 (CNN)	2,158,210	2,158,210	19,993,888

These FLOPs values represent the approximate number of floating-point operations required for a forward pass through the respective CNN models. Please note that these values are estimates and can vary depending on the specific implementation, framework, and optimizations used. It's important to mention that FLOPs can be influenced by factors like input image size, batch size, and device type. The values provided above assume a standard input size and batch size on a GPU with Tensor Cores.
the complexity of the Simplest color balance is also calculated; Based on the Algorithm of Simplest Color Balance, we can estimate the complexity of the algorithm:

- The loop iterates over the channels of the image, so its complexity depends on the number of channels.

- The histogram calculation and cumulative sum have a complexity of O(n), where n is the number of pixels in each channel.
- The search sorted function has a complexity of O(log n), where n is the number of bins in the histogram (256 in this case).
- The construction of the lookup table has a complexity of O(1) since it operates on a fixed number of values (255).
- The contrast adjustment using LUT has a complexity of O(n), where n is the number of pixels in each channel.

Therefore, the overall complexity of the algorithm can be approximated as **O(c * n * log n)**, where c=1 is the number of channels and n is the number of pixels in each channel.

Accuracy and Loss: we have used the test dataset to validate the models.
In our model, we got an accuracy of 98 %, and a Loss rate of 0.019 for the proposed Model using an early stopping option with a 0.00001 learning rate, for the same learning rate, and with 120 epochs the accuracy reaches 100% when the loss is 0.004. for the proposed Model using an early stopping option with a 0.001 learning rate, the accuracy is 93 %, and a Loss rate of 0.139, for the same learning rate, and with 120 epochs the accuracy reaches 98% when the loss is 0.006. those results outperform all the concurrent models especially when we used the 120 epoch option. figure 3.29 represents the model's training and validation accuracy and loss curves and figure 3.30 represents the ROC Curve.
In VGG16 we got an accuracy of 98.99 %, a loss rate of 0.006 in the option of No weight used, and a learning rate of 0.00001 and No freezing of layers. the others were far below our performances.
tables 3.12, 3.13, 3.14,3.15, and 3.16 even 3.17 represent the metrics of the models on five external datasets.

3.6.5 Generalization of results on external Datasets

The proposed approach was compared to five deep learning methods to assess its effectiveness and generalizability to other datasets not used in the training process, namely: Inception V3, VGG 16, VGG 19, ResNet, and DenseNet. Each model was implemented and tested with eight possible configurations by enabling/disabling the following parameters: Weight, Learning Rate, and Freeze.

N°	Dataset	COVID-19 Radiography Database		Covid-19 Image Dataset database		Chest X-ray for covid-19 detection database		COVID-19 chest x-ray database		Covid Patients Chest x-ray database	
	Total images	2592		222		313		356		162	
	The Prediction	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP
1	our model LR 120 epochs	2526	66	178	44	304	9	332	24	157	5
2	our model NOLR 120 epochs	2137	455	166	56	264	49	308	48	162	0
3	our model LR earlystop	2462	130	193	29	299	14	294	62	136	26
4	our model NOLR earlystop	1998	594	140	82	254	59	143	213	62	100
5	InceptionV3_W_LR-freeze	1212	1380	132	90	144	169	356	0	162	0
6	InceptionV3_W_NOLR-freeze	1212	1380	132	90	144	169	356	0	162	0
7	InceptionV3_W_LR-NOfreeze	1212	1380	132	90	144	169	356	0	162	0
8	InceptionV3_W_NOLR-NOfreeze	1212	1380	132	90	144	169	356	0	162	0
9	InceptionV3_NOW_LR-freeze	1458	1134	102	120	176	137	17	339	3	159
10	InceptionV3_NOW_NOLR-freeze	2268	324	191	31	285	28	234	122	110	52
11	InceptionV3_NOW_LR-NOfreeze	1212	1380	132	90	144	169	356	0	162	0
12	InceptionV3_NOW_NOLR-NOfreeze	1212	1380	132	90	144	169	356	0	162	0
13	VGG16_W_LR-freeze	1388	1204	95	127	171	142	1	355	0	162
14	VGG16_W_NOLR-freeze	2395	197	164	58	262	51	258	98	130	32
15	VGG16_W_LR-NOfreeze	2511	81	182	40	300	13	354	2	162	0
16	VGG16_W_NOLR-NOfreeze	1101	1491	111	111	193	120	163	193	68	94
17	VGG16_NOW_LR-freeze	1380	1212	91	131	169	144	0	356	0	162
18	VGG16_NOW_NOLR-freeze	2269	323	181	41	132	181	249	107	116	46
19	VGG16_NOW_LR-NOfreeze	2356	236	155	67	303	10	277	79	130	32
20	VGG16_NOW_NOLR-NOfreeze	1430	1162	93	129	171	142	22	334	3	159
21	VGG19_W_LR-freeze	1737	857	116	106	202	111	101	255	123	39
22	VGG19_W_NOLR-freeze	1783	809	121	101	220	93	119	237	46	116
23	VGG19_W_LR-NOfreeze	1212	1380	132	90	144	169	356	0	162	0

24	*VGG19_W_NOLR-NOfreeze*	*1383*	*1209*	*92*	*130*	*169*	*144*	*5*	*351*	*0*	*162*
25	*VGG19_NOW_LR-freeze*	*1380*	*1212*	*90*	*132*	*169*	*144*	*0*	*356*	*0*	*162*
26	*VGG19_NOW_NOLR-freeze*	*2210*	*382*	*178*	*44*	*298*	*15*	*229*	*127*	*110*	*52*
27	*VGG19_NOW_LR-NOfreeze*	*2453*	*139*	*176*	*46*	*146*	*167*	*309*	*47*	*147*	*15*
28	*VGG19_NOW_NOLR-NOfreeze*	*1211*	*1381*	*132*	*90*	*144*	*169*	*356*	*0*	*162*	*0*
29	*ResNet_W_LR-freeze*	*1253*	*1339*	*132*	*90*	*151*	*162*	*340*	*16*	*162*	*1*
30	*ResNet_W_NOLR-freeze*	*1212*	*1380*	*132*	*90*	*144*	*169*	*356*	*0*	*162*	*0*
31	*ResNet_W_LR-NOfreeze*	*1212*	*1380*	*132*	*90*	*144*	*169*	*356*	*0*	*162*	*0*
32	*ResNet_W_NOLR-NOfreeze*	*1212*	*1380*	*132*	*90*	*144*	*169*	*356*	*0*	*162*	*0*
33	*ResNet_NOW_LR-freeze*	*2335*	*257*	*184*	*38*	*297*	*16*	*272*	*84*	*126*	*36*
34	*ResNet_NOW_NOLR-freeze*	*2457*	*135*	*182*	*40*	*302*	*11*	*308*	*48*	*147*	*15*
35	*ResNet_NOW_LR-NOfreeze*	*1212*	*1380*	*132*	*90*	*144*	*169*	*356*	*0*	*162*	*0*
36	*ResNet_NOW_NOLR-NOfreeze*	*1212*	*1380*	*132*	*90*	*144*	*169*	*356*	*0*	*162*	*0*
37	*DenseNet_W_LR-NOfreeze*	*1212*	*1380*	*132*	*90*	*144*	*169*	*356*	*0*	*162*	*0*
38	*DenseNet_W_LR-freeze*	*1212*	*1380*	*129*	*93*	*144*	*169*	*349*	*7*	*162*	*0*
39	*DenseNet_W_NOLR-freeze*	*1210*	*1382*	*133*	*89*	*144*	*169*	*349*	*7*	*161*	*1*
40	*DenseNet_W_NOLR-NOfreeze*	*1213*	*1379*	*134*	*88*	*148*	*165*	350	6	162	0
41	*DenseNet_NOW_LR-NOfreeze*	*2302*	*290*	*175*	*47*	*296*	*17*	*262*	*94*	*129*	*33*
42	*DenseNet_NOW_LR-freeze*	*1212*	*1380*	*132*	*90*	*144*	*169*	*356*	*0*	*162*	*0*
43	*DenseNet_NOW_NOLR-NOfreeze*	*2475*	*117*	*180*	*42*	*304*	*9*	308	48	150	12
44	*DenseNet_NOW_NOLR-freeze*	*1212*	*1380*	*132*	*90*	*144*	*169*	*356*	*0*	*162*	*0*

Tableau 3.11: globals results

thos following (Figures : 3.35, 3.33, 3.36 , 3.34 , 3.37) shows graphically the given results on every single dataset of all the models including our models.

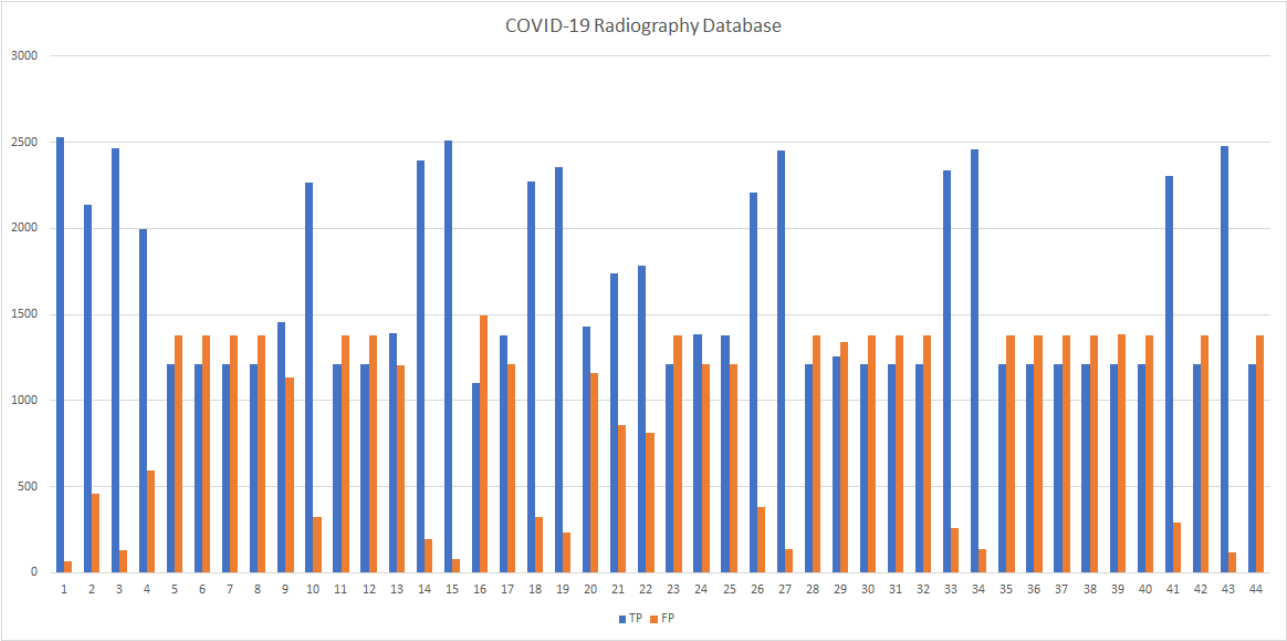


Figure 3.33: external results graph on COVID-19 RADIOGRAPHY DATABASE [14]

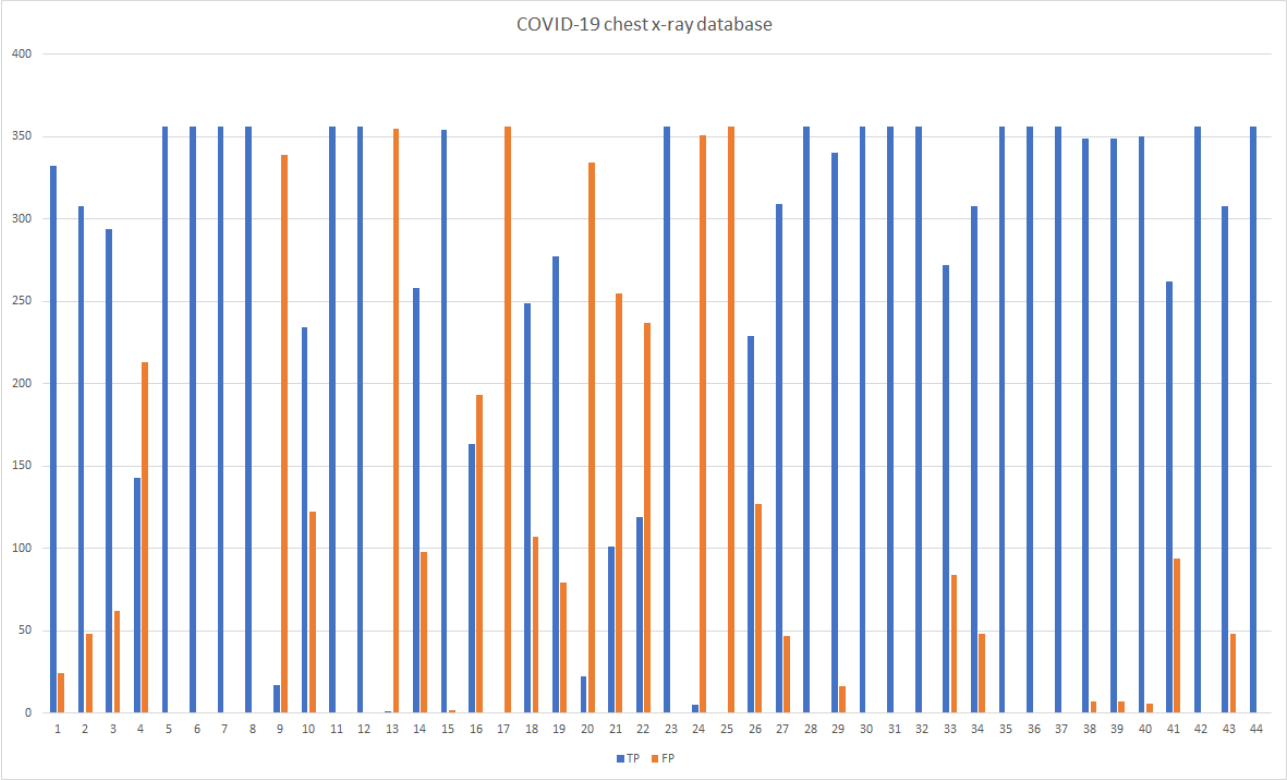


Figure 3.34: external results graph on COVID-19 chest x-ray dataset [15]

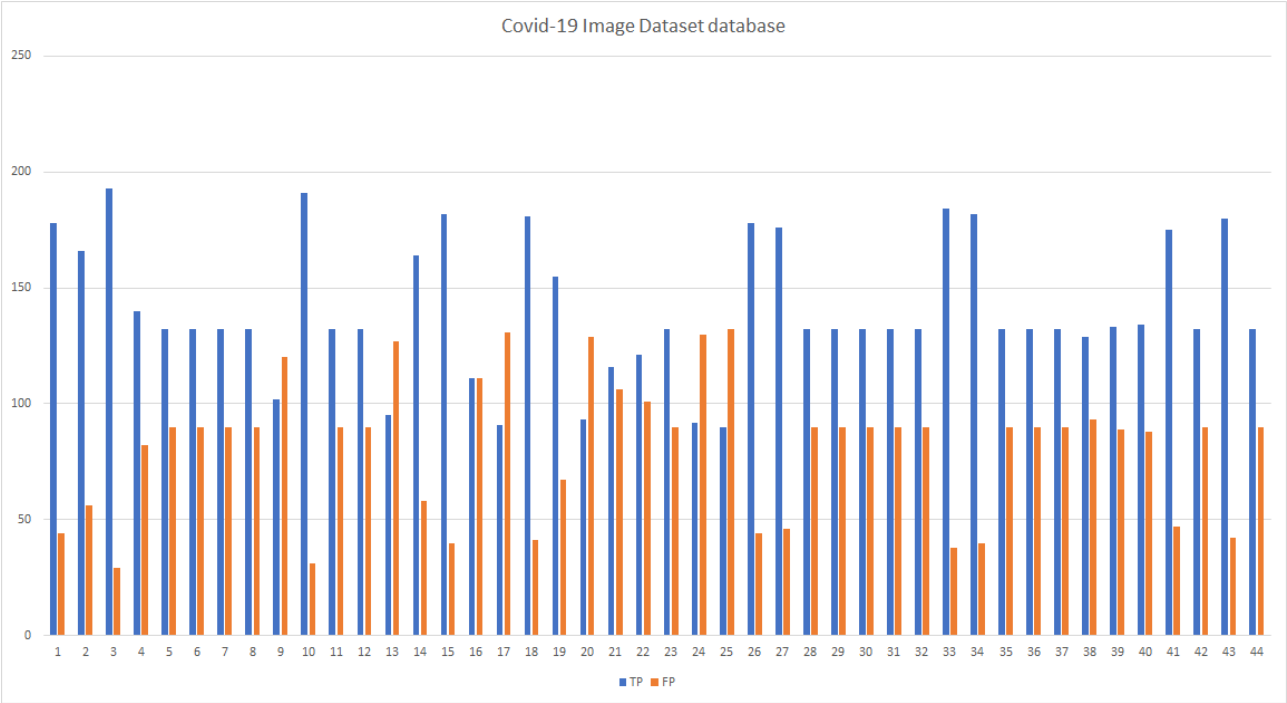


Figure 3.35: external results graph on COVID-19 Image Dataset [16]

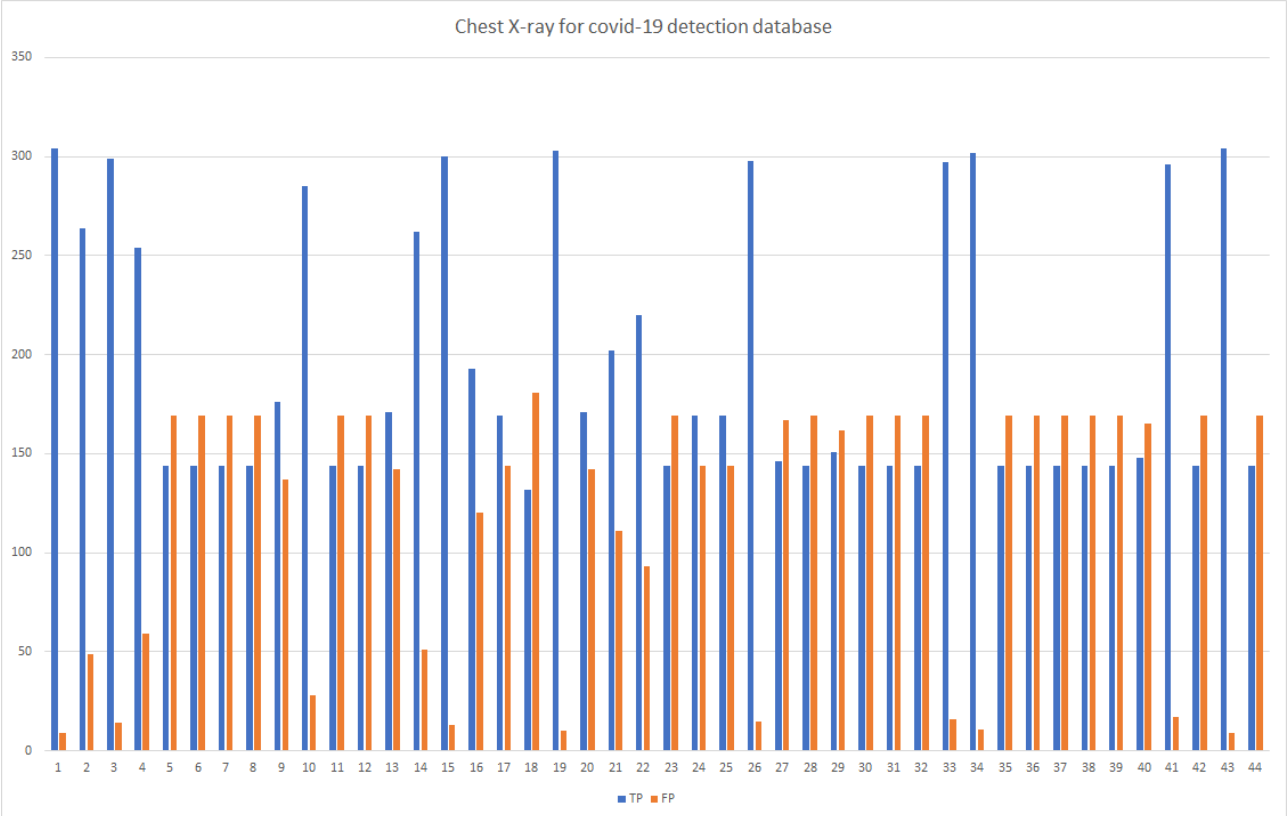


Figure 3.36: external results graph on Chest X-ray for COVID-19 detection dataset [17]

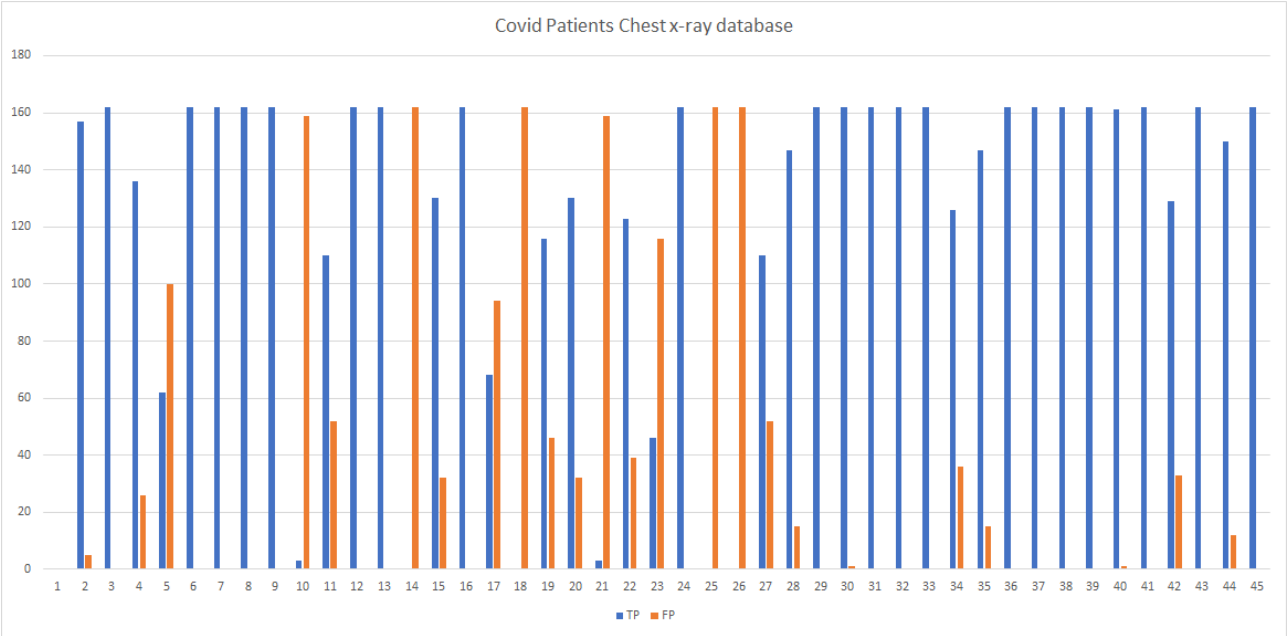


Figure 3.37: external results graph on Covid Patients Chest x-ray dataset [18]

We have observed in the statistic figures of all the best models (Figures : 3.35, 3.33, 3.36 , 3.34 , 3.37) that our model consistently delivers excellent results across various datasets and scenarios. In contrast, other models may perform well in one dataset but falter in another. Our models consistently achieve superior metrics and accuracy while requiring minimal memory and time consumption.

Tables 3.12, 3.13, 3.14,3.15, and 3.16 present the results obtained from "only" the best configuration of each model Precision, accuracy, loss, and ROC metrics are calculated on five challenging datasets which are also described in the Table 3.2.

The First Dataset: is COVID-19 RADIOGRAPHY DATABASE [14] (Winner of the COVID-19 Dataset Award by Kaggle Community), was created by a team of researchers in collaboration with medical doctors, and it's Academic/Non-Commercial data. All the chest X-ray images for COVID-19 positive cases (were tested COVID positive with PCR or CT), along with Normal and Viral Pneumonia images [177]. This dataset was released in stages (many updates), and mainly come from 5 open-source datasets: the Italian Society of Medical and Interventional Radiology (SIRM) was used for the COVID-19 Chest X-Ray dataset [178], Valencia Region Image Bank (BIMCV) padchest dataset [179] and J. P. Cohen [19]. Finally, for the Normal images the Chest X-Ray Images (pneumonia) database in Kaggle [168] and Radiological Society of North America (RSNA) Kaggle database were adopted [180]. After cleaning and keeping only frontale and x-ray images, only two classes from this database was taken, the Covid class contains 1212 photos of people infected with Covid-19 disease, and the second class of normal people, which contains 1380 photos. All the images are in Portable Network Graphics (PNG) file format.

Tableau 3.12: Results on COVID19 Radiography Database [14]

Model Name	Parameters	Proposed	InceptionV3	VGG16	VGG19	ResNet	DenseNet
Best	Weight	x	x	✓	x	x	x
	Learning Rate	✓	x	✓	✓	x	x
Configuration	Freeze	x	✓	x	x	✓	x
Results	TP	**97.45**	87.5	96.88	94.64	94.79	95,49
	FP	**2,55**	12,5	3,13	5,36	5,21	4,51
	Precision	97.96	83.57	90.22	94.40	85.07	94.62
	Recall	97.32	99.13	91.52	96.44	98.69	97.03
	accuracy	97.50	89.16	95.06	95.06	90.08	95.48
	loss	0.009	0.211	0.25	0.021	0.138	0.023
	ROC	0.98	0.88	0.95	0.95	0.89	0.95

The Second Dataset: This database (COVID-19 chest x-ray dataset) [15] of COVID-19 cases with chest X-ray or CT images. It contains COVID-19 cases as well as MERS, SARS, and ARDS. We kept only the covid class containing 357 photos of people with Covid-19 disease, they were combined between confirmed cases and not confirmed by pcr, and CT scan.

In the table 3.13, because we had only one class of COVID-19 cases and No Normal class, So the TP and FP are very informative.

Tableau 3.13: Results on COVID-19 chest x-ray dataset [15]

Model Name	Parameters	Proposed	InceptionV3	VGG16	VGG19	ResNet	DenseNet
Best	Weight	x	x	✓	x	✓	x
	Learning Rate	✓	x	✓	✓	✓	x
Configuration	Freeze	x	✓	x	x	✓	x
Results	TP	93,26	65,73	**99,44**	86,8	95,51	86,52
	FP	6,74	34,27	**0,56**	13,2	4,49	13,48

The Third Dataset: This database [16] contains a total of 317 photos of people with lung disease and Covid and chest photos of normal people all x-ray images in *.png, *.jpg format. The Lung Disease class contains 90 images, and the Covid class contains 137 images and the Normal People class contains 90, we kept the covid and normal classes. in this dataset it wasn't mentioned any confirmation by PCR was used.

Tableau 3.14: Results on COVID-19 Image Dataset [16]

Model Name	Parameters	Proposed	InceptionV3	VGG16	VGG19	ResNet	DenseNet
Best	Weight	x	x	✓	x	x	x
	Learning Rate	✓	x	✓	x	✓	x
Configuration	Freeze	x	✓	x	✓	✓	x
Results	TP	**86,94**	86,04	81,98	80,18	82,88	81,08
	FP	**13,06**	13,96	18,02	19,82	17,12	18,92
	Precision	95.83	79.95	78.50	77.34	80.13	79.10
	Recall	74.44	86.66	72.02	71.12	73.50	73.56
	accuracy	86.93	85.66	81.98	79.23	81.17	81.88
	loss	0.014	0.276	0.092	0.231	0.194	0.104
	ROC	0.86	0.86	0.85	0.84	0.85	0.85

The Forth Dataset: This database contains 371 images and is divided into two classes "Covid" and "Normal", each class contains approximately 174 images. All files are in jpeg/jpg/png formate. They was collected from two big datasets : covid patient's images from and normal patient's images from Chest X-Ray Images (Pneumonia) [168];

Tableau 3.15: Results on Chest X-ray for COVID-19 detection dataset [17]

Model Name	Parameters	Proposed	InceptionV3	VGG16	VGG19	ResNet	DenseNet
Best	Weight	x	x	x	x	x	x
	Learning Rate	✓	x	✓	x	x	x
Configuration	Freeze	x	✓	x	✓	✓	x
Results	TP	**97,12**	91,05	96,81	95,21	96,49	**97,12**
	FP	**2,88**	8,95	3,19	4,79	3,51	**2,88**
	Precision	98.78	88.88	97.00	93.85	98.17	98.78
	Recall	97.63	99.40	96.16	99.40	95.26	95.85
	accuracy	97.12	92.97	96.16	96.16	96.48	97.12
	loss	0.010	0.212	0.022	0.268	0.005	0.017
	ROC	0.97	0.92	0.96	0.96	0.97	0.97

The Fifth Dataset: In order to further ensure the quality of the models, we also tested it on the covid class of this database [18] called Covid Patients Chest x-ray dataset, it contains 162 images of lung x-ray of covid people and 162 images of lung x-ray of normal people out of a total of 324. These images have not been verified by PCR either.

Tableau 3.16: Results on Covid Patients Chest x-ray dataset [18]

Model Name	Parameters	Proposed	InceptionV3	VGG16	VGG19	ResNet	DenseNet
Best	Weight	x	x	✓	x	✓	x
	Learning Rate	x	x	✓	✓	✓	x
Configuration	Freeze	x	✓	x	x	✓	x
Results	TP	**100**	67,9	**100**	90,74	99,39	92,59
	FP	**0**	32,1	**0**	9,26	0,61	7,41
	Precision	95.83	79.59	97.50	75.25	80.23	94.44
	Recall	74.44	86.66	43.33	81.11	76.66	56.66
	accuracy	86.93	85.85	76.57	81.53	82.88	81.08
	loss	0.014	0.276	0.022	0.367	0.210	0.028
	ROC	0.85	0.82	0.71	0.81	0.82	0.77

The proposed model ranks first in four datasets and third in the others and surpassed them in many evaluation metrics, even though only the best configuration of the other methods was used in the comparison. In addition, the tables show that DenseNet, VGG16, and InceptionV3 maintained the same best configuration across all datasets. Thus, in most cases, the proposed approach can solve the generalization problem and go beyond the most popular methods.

Table 3.17 quantitative representation of the Matrix of confusion (TP and FP) results obtained from the best configuration of each model calculated on the same challenging datasets.

The use of TP and FP of the best hyperparameters options of the five deep learning methods in the generalization on external bases is enough since they are representative and because most of the other metrics (Precision, Recall, F1-Score, Accuracy, Loss, and ROC).

Tableau 3.17: Comparative results on five public datasets
True Prediction(TP) False prediction(FP), W/NOW: Weight/No Weight LR/NOLR: Learning Rate/ No Learning Rate

Dataset	COVID-19 Radiography Dataset		Covid-19 Image Dataset		Chest X-ray for covid-19 detection dataset		COVID-19 chest x-ray Dataset		Covid Patients Chest x-ray Dataset	
Prediction (%)	TP %	FP %	TP %	FP %	TP %	FP %	TP %	FP %	TP %	FP %
Proposed model LR 120 epochs	97,45	2,55	80,18	19,82	97,12	2,88	93,26	6,74	96,91	96,91
Proposed model NOLR/ 120 epochs	82,45	17,55	74,77	25,23	84,35	15,65	86,52	13,48	100	0
Proposed model LR/ earlystop	94,98	5,02	86,94	13,06	95,53	4,47	82,58	17,42	83,95	16,05
Proposed model NOLR/ earlystop	77,08	22,92	63,06	36,94	81,15	18,85	40,17	59,83	38,27	61,73
Inception V3 NOW/LR/freeze	56,25	43,75	45,95	54,05	56,23	43,77	4,78	95,22	1,85	98,15
Inception V3 NOW/NOLR /freeze	87,5	12,5	86,04	13,96	91,05	8,95	65,73	34,27	67,9	32,1
VGG16 W/NOLR/freeze	92,4	7,6	73,87	26,13	83,71	16,29	72,47	27,53	80,25	19,75
VGG16 W/LR/NOfreeze	96,88	3,13	81,98	18,02	95,85	4,15	99,44	0,56	100	0
VGG16 W/NOLR/NOfreeze	42,48	57,52	50	50	61,66	38,34	45,79	54,21	41,98	58,02
VGG16 NOW/NOLR/freeze	87,54	12,46	81,53	18,47	42,17	57,83	69,94	30,06	71,6	28,4
VGG16 NOW/LR/NOfreeze	90,9	9,1	69,82	30,18	96,81	3,19	77,81	22,19	80,25	19,75
VGG19 NOW/NOLR/freeze	66,96	33,04	52,25	47,75	64,54	35,46	28,37	71,63	75,93	24,07
VGG19 W/NOLR/freeze	68,79	31,21	54,5	45,5	70,29	29,71	33,43	66,57	28,4	71,6
VGG19 NOW/NOLR/freeze	85,26	14,74	80,18	19,82	95,21	4,79	64,33	35,67	67,9	32,1
VGG19 NOW/LR/NOfreeze	94,64	5,36	79,28	20,72	46,65	53,35	86,8	13,2	90,74	9,26
ResNet W/LR/freeze	48,34	51,66	59,46	40,54	48,24	51,76	95,51	4,49	99,39	0,61
ResNet NOW/LR/freeze	90,08	9,92	82,88	17,12	94,89	5,11	76,4	23,6	77,78	22,22
ResNet NOW/NOLR/freeze	94,79	5,21	81,98	18,02	96,49	3,51	86,52	13,48	90,74	9,26
DenseNet NOW/LR/NOfreeze	88,81	11,19	78,83	21,17	94,57	5,43	73,6	26,4	79,63	20,37
DenseNet NOW/NOLR/NOfreeze	95,49	4,51	81,08	18,92	97,12	2,88	86,52	13,48	92,59	7,41

Furthermore, despite the addition of the preprocessing step, the processing time was not affected, and the proposed COVID-19 detection system still works in real time. The time required to classify an X-ray image is shown in Table 3.18, it was calculated in a CPU and a GPU, we took the average time for all the processes applied on all the used datasets, and we calculated the Standard deviation to know the range of the time, so the time of each operation can be in the range of $[mean - std, mean + std]$.

Tableau 3.18: Processing Time

Operation	***Time CPU(s)***		***Time GPU(s)***	
	mean	std	mean	std
Color Balance	0.002	0.001	0.001	0.001
Lungs segmentation	2.412	0.962	0.116	0.052
Classification	0.279	0.099	0.075	0.035
Total Time	2.693	1.062	0.192	0.088

To justify that our solution is more efficient, we have calculated the total Time on CPU and GPU, to prove that our approach can be used in an embedded system like RPi4 or another and in real-time since the FPS and the processing time is acceptable for an image of 512x512 resolution. and the pre-processing steps are low-consuming time compared to their benefits for the global system's accuracy.
So the system can be used by doctors with phones or raspberries in real time to detect, segment, and classify lungs infected with covid-19.

we have calculated also the complexity, FLOPs and parameter size metrics to calculate the size, weight and computational consumption of our CNN and the used pre-processing steps compared to the other models in table 3.10. and ours is still low consuming in time and resources.

3.7 Conclusion

In conclusion, radiological imaging stands out as a promising avenue for COVID-19 detection due to its speed and cost-effectiveness compared to traditional methods like RT-PCR. While deep learning-based approaches have been explored extensively, their performance often falters when applied to new datasets due to limited training data availability. Addressing this challenge, this chapter presented a novel CNN-based approach proposed for COVID-19 detection in radiological images.

The proposed method focuses on simplifying the deep learning model's architecture to accommodate limited training data, supplemented by a preprocessing step aimed at refining image quality and extracting the relevant lung regions. Through a comprehensive three-stage lung segmentation process, including noise reduction, color balance correction, UNet-based lung segmentation, and data augmentation, the proposed approach effectively enhances the training

dataset and model robustness. Followed be a fourth satge of classification using a light weight CNN to train a well constructed dataset.

The proposed preprocessing method allows: First, the images were improved in quality and any noise or artifacts were removed. Second, to improve the image, a color balance correction was performed. The region of interest was then extracted using a UNet segmentation of the lungs in the image. Finally, image data augmentation was used to increase the size of the training dataset and improve the model's robustness. The CNN model was trained using two merged datasets and tested with four different configurations using well-known metrics like F1-sore, recall, precision, and accuracy. The best of them scored 100% on almost every metric.

Evaluation of the proposed method against established deep learning models, such as Inception v3, ResNet50, DenseNet, VGG16, and VGG19, (Each one has eight different possible configurations) on multiple datasets demonstrates its superior performance across various metrics. Notably, our method achieves remarkable scores, highlighting its efficacy in COVID-19 detection.

Despite the rigorous experimentation and optimization conducted in this study, we acknowledge certain limitations, particularly the focus solely on classification rather than object detection. Future research directions include expanding the method's scope to encompass other pneumonia types, implementing object detection algorithms like YOLO, MaskRCNN, or RetinaNet for precise lung detection in video streams, and exploring deployment on embedded systems for real-world application.

In summary, this research presents a promising framework for COVID-19 detection in radiological images, laying the groundwork for future advancements in pneumonia diagnosis and real-time medical imaging solutions.

General Conclusion

The detection of COVID-19 in X-ray images presents a critical and challenging problem with significant implications for healthcare systems worldwide. The COVID-19 pandemic has underscored the urgent need for faster, more accessible diagnostic methods to aid in reducing mortality rates and preventing disease transmission. Traditional diagnostic methods like RT-PCR tests, while reliable, are hindered by factors such as time consumption, costliness, and the risk of exposure for healthcare workers. In response, medical imaging techniques such as X-rays, coupled with artificial intelligence (AI) through convolutional neural networks (CNNs), have emerged as valuable tools for COVID-19 screening.

This thesis presented a novel approach to COVID-19 diagnosis in X-ray images, leveraging deep learning techniques to develop a lightweight CNN optimized for small training datasets and low computation cost. The proposed solution included the development of a custom dataset comprising confirmed COVID-19 subclasses, preprocessing techniques to enhance image quality, lung segmentation to eliminate irrelevant data, and data augmentation to address class imbalance. Extensive testing across various external datasets has demonstrated the real-time capability, reliability, and superior performance of the proposed model compared to traditional CNN models.

Our solution is designed to be affordable, portable, and efficient, making it suitable for everyday use. doctors can use it online or offline, in a desktop or embedded system like RPi4, and using scanned x-ray images or flow camera, through webcam and lungs detection and segmentation to a cropped ROI to be classified as COVID or normal.
Second the proposed approach tends to enhance several steps: from the good light and gray/chrominance balancing to the segmentation of a region of interest to avoid any miss leading information and to augment the accuracy, using UNET deeplearning based segmentation followed by a lightweight model to classify the existence of COVID after trying several benchmark CNNs that proved their efficiency in the field of objects classification and tried so many options of hyperparameters.
to come to the end with the best architecture that fits this kind of problem.
Some extra metrics, analyses, and discussions are added to the manuscript to make it easy to understand from the doctors or any interested person on this field. even if some persons may say that COVID-19 is over:

0. despite the concern, of being out of date because we think the pandemic is over. we draw the attention that COVID-19 is still a danger since we are not in the zero case

situation. The statistics of the world health organisation are saying that 20,690,872 Currently Infected Patients, 20,653,705 (99.8%) in Mild Condition, and 37,167 (0.2%) Serious or Critical. We are happy that the number of critical cases is decreasing but we have to be in guard of any rise of numbers.

0. Even though the situation is very stable, this thesis can be a starting point for any other similar situation of classification of any lung disease using very few amount of data, using many options (transfer learning, fine tuning, bottleneck learning, or lightweight model with some pre-processing steps to leverage the relevant data by keeping only ROI and carefully enhanced content).

And some persons may say that PCR test and vaccination are available and no need to a CNN based Solutions:

0. we can say that despite the fact that PCRs are available now, but they are still costing money and take much time to give results beside the errors, so the alternative of a faster and more accurate Artificial intelligence based system to help the doctors to decide is always welcome.

Our system was extensively tested with different model's configurations and in various scenarios to evaluate its performance and reliability. The results showed that our solution is capable of accurately detecting COVID-19 in real-time and providing a good solution for embedded systems. We also demonstrated that our system is highly portable and efficient, making it a practical and effective solution for medical diagnosis.

Finally, this study offers a comprehensive and innovative solution to the challenge of COVID-19 detection in X-ray images, contributing to the advancement of medical decision support systems in the fight against the pandemic, particularly in resource-constrained regions. By providing a faster, more accessible, and efficient method for COVID-19 screening, this research has the potential to significantly impact healthcare outcomes and contribute to the global effort to combat the spread of the virus.

Perspectives: Even though we tried to enhance every step from the choice of data to the selection of confirmed data, to the experimentation on several benchmarks models, to the selection of preprocessing and segmentation, and the design of a new simple light CNN and tuning the parameters and generalizing on real world cases and low consuming time and resources.

We have some limitations:

0. We didn't treat the percentage of ill tissues, and the solution is to use the segmentation algorithm UNet or MaskRCNN, to calculate the MeanIOU.

0. we didn't treat the situation of classifying many types of pneumonia diseases, and the solution is first of all to collect labeled data and to use a deeper CNN.

0. We treated the problem only as a classification problem, but we could use some object detection algorithms because the objective is to help doctors to use adequate devices by scanning X-ray images, that's why the system must detect lungs in the video flow before classifying it.

0.

Bibliography

[1] Colin E Willoughby, Diego Ponzin, Stefano Ferrari, Aires Lobo, Klara Landau, and Yadollah Omidi. Anatomy and physiology of the human eye: effects of mucopolysaccharidoses disease on structure and function–a review. *Clinical and Experimental Ophthalmology*, 38:2–11, 2010.

[2] Andrej Karpathy. *Neural Networks Part 1: Setting Up the Architecture. Notes for CS231n Convolutional Neural Networks for Visual Recognition*. Stanford University: Stanford, CA, USA, 2016.

[3] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

[4] Ting Yuan, Lin Lv, Fan Zhang, Jun Fu, Jin Gao, Junxiong Zhang, Wei Li, Chunlong Zhang, and Wenqiang Zhang. Robust cherry tomatoes detection algorithm in greenhouse scene based on ssd. *Agriculture*, 10:160, 05 2020.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. 2020.

[6] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[7] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[8] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. July 2017.

[9] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[10] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, Yiduo Li, Bo Zhang, Yufei Liang, Linyuan Zhou, Xiaoming Xu, Xiangxiang Chu, Xiaoming Wei, and Xiaolin Wei. Yolov6: A single-stage object detection framework for industrial applications, 2022.

[11] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022.

[12] Juan Terven and Diana Cordova-Esparza. A comprehensive review of yolo: From yolov1 and beyond, 2023.

[13] Shay Aharon, Louis-Dupont, Ofri Masad, Kate Yurkova, Lotem Fridman, Lkdci, Eugene Khvedchenya, Ran Rubin, Natan Bagrov, Borys Tymchenko, Tomer Keren, Alexander Zhilko, and Eran-Deci. Super-gradients, 2021.

[14] Muhammad EH Chowdhury, Tawsifur Rahman, Amith Khandakar, Rashid Mazhar, Muhammad Abdul Kadir, Zaid Bin Mahbub, Khandakar Reajul Islam, Muhammad Salman Khan, Atif Iqbal, Nasser Al Emadi, et al. Can ai help in screening viral and covid-19 pneumonia? *Ieee Access*, 8:132665–132676, 2020.

[15] Bachir. covid-chest-xray, 2020. Online.

[16] Pranav Raikote. Covid-19 image dataset, 2020. Online.

[17] Fusic Fenta. Chest xray for covid-19 detection, 2020. Online.

[18] Ankita Choudhury. covid patients chest xray, 2021. Online.

[19] Joseph Paul Cohen, Paul Morrison, and Lan Dao. Covid-19 image data collection. *arXiv 2003.11597*, 2020.

[20] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition, 2010.

[21] E. Alpaydin. *Introduction to Machine Learning, second edition*. Adaptive Computation and Machine Learning series. MIT Press, 2009. Accessed: 2023-04-20.

[22] Jeff Heaton. Ian goodfellow, yoshua bengio, and aaron courville: Deep learning. *Genetic Programming and Evolvable Machines*, 19(1):305–307, Jun 2018. Accessed: 2023-04-20.

[23] T Mitchell. *Machine Learning*. McGraw Hill, 1997. Accessed: 2023-04-20.

[24] Uzair Aslam Bhatti, Hao Tang, Guilu Wu, Shah Marjan, and Aamir Hussain. Deep learning with graph convolutional networks: An overview and latest applications in computational intelligence, Feb 2023.

[25] Nur Indah Lestari, Walayat Hussain, Jose M. Merigo, and Mahmoud Bekhit. A survey of trendy financial sector applications of machine and deep learning. *Springer Nature Switzerland*, pages 619–633, 2023. Accessed: 2023-04-20.

[26] Xue-Wen Chen and Xiaotong Lin. Big data deep learning: Challenges and perspectives. *IEEE Access*, 2:514–525, 2014.

[27] Goodfellow Ian, Bengio Yoshua, and Courville Aaron. *Deep Learning*. MIT Press, 2016.

[28] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[29] Robin M Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. *arXiv preprint arXiv:1912.05911*, 2019.

[30] Divya Saxena and Jiannong Cao. Generative adversarial networks (gans) challenges, solutions, and future directions. *ACM Computing Surveys (CSUR)*, 54(3):1–42, 2021.

[31] Zihan Ding, Yanhua Huang, Hang Yuan, and Hao Dong. Introduction to reinforcement learning. *Deep Reinforcement Learning: Fundamentals, Research and Applications*, pages 47–123, 2020.

[32] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021.

[33] Tingxi Wen and Zhongnan Zhang. Deep convolution neural network and autoencoders-based unsupervised feature learning of eeg signals. *IEEE Access*, 6:25399–25410, 2018.

[34] Xiongwei Wu, Doyen Sahoo, and Steven CH Hoi. Recent advances in deep learning for object detection. *Neurocomputing*, 396:39–64, 2020.

[35] Görkem Algan and Ilkay Ulusoy. Image classification with deep learning in the presence of noisy labels: A survey. *Knowledge-Based Systems*, 215:106771, 2021.

[36] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3523–3542, 2022.

[37] Asyari Muhammad Zacky, Filbert Sebastian, and Sukra Zener Lie". Histogram of oriented gradients (hog) and haar cascade with convolutional neural network (cnn) performance comparison in the application of edge home security system. In Subhas Chandra Mukhopadhyay, S.M. Namal Arosha Senanayake, and P.W. Chandana Withana, editors, *Innovative Technologies in Intelligent Systems and Industrial Applications*, pages 13–22, Cham, 2023. Springer Nature Switzerland.

[38] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

[39] Ross Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[40] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[41] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[42] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[43] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection, 2020.

[44] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[45] Seyed Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian D. Reid, and Silvio Savarese. Generalized intersection over union: A metric and A loss for bounding box regression. *CoRR*, abs/1902.09630, 2019.

[46] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[47] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020.

[48] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, Ayush Chaurasia, TaoXie, Liu Changyu, Abhiram V, tkianai, yxNONG, Adam Hogan, lorenzomammana, AlexWang1900, Jan Hajek, Laurentiu Diaconu, Marc, Yonghye Kwon, oleg, and Francisco Ingham. ultralytics/yolov5: v5.0 - yolov5-p6 1280 models, aws, supervisely and youtube integrations (v5.0).

[49] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023. online.

[50] glenn jocher. Yolo-nas. https://docs.ultralytics.com/fr/models/yolo-nas/. online.

[51] MS COCO (Microsoft Common Objects in Context)dataset. https://cocodataset.org/. Accessed: 2023-04-30.

[52] PASCAL VOC (ual Object Classes))dataset. http://host.robots.ox.ac.uk/pascal/VOC/. Accessed: 2023-04-30.

[53] Open Images Dataset V7dataset. https://storage.googleapis.com/openimages/web/index.html. Accessed: 2023-05-02.

[54] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[55] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[56] Abien Fred Agarap. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375, 2018.

[57] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118, 2010.

[58] Kyle Matoba, Nikolaos Dimitriadis, and François Fleuret. The theoretical expressiveness of maxpooling, 2022.

[59] R. Lokesh Kumar, Jagadeesh Kakarla, B. Venkateswarlu Isunuri, and Munesh Singh. Multi-class brain tumor classification using residual network and global average pooling. *Multimedia Tools and Applications*, 80(9):13429–13438, Apr 2021.

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014.

[61] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. *CoRR*, abs/1904.08189, 2019.

[62] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[63] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[64] Yunji Chen, Ling Li, Wei Li, Qi Guo, Zidong Du, and Zichen Xu. Chapter 2 - fundamentals of neural networks. In Yunji Chen, Ling Li, Wei Li, Qi Guo, Zidong Du, and Zichen Xu, editors, *AI Computing Systems*, pages 17–51. Morgan Kaufmann, 2024.

[65] Kai Lv, Shuo Zhang, Tianle Gu, Shuhao Xing, Jiawei Hong, Keyu Chen, Xiaoran Liu, Yuqing Yang, Honglin Guo, Tengxiao Liu, Yu Sun, Qipeng Guo, Hang Yan, and Xipeng Qiu. Collie: Collaborative training of large language models in an efficient way, 2023.

[66] Hang Xu, Wenxuan Zhang, Jiawei Fei, Yuzhe Wu, Tingwen Xie, Jun Huang, Yuchen Xie, Mohamed Elhoseiny, and Panos Kalnis. SLAMB: Accelerated large batch training with sparse communication. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 38801–38825. PMLR, 23–29 Jul 2023.

[67] Aochuan Chen, Yimeng Zhang, Jinghan Jia, James Diffenderfer, Jiancheng Liu, Konstantinos Parasyris, Yihua Zhang, Zheng Zhang, Bhavya Kailkhura, and Sijia Liu. Deepzero: Scaling up zeroth-order optimization for deep model training, 2024.

[68] Atılım Güneş Baydin, Barak A. Pearlmutter, Don Syme, Frank Wood, and Philip Torr. Gradients without backpropagation, 2022.

[69] Jun Lu. Adasmooth: An adaptive learning rate method based on effective ratio, 2022.

[70] Hanbaek Lyu. Stochastic regularized majorization-minimization with weakly convex and multi-convex surrogates, 2023.

[71] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[72] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[73] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost, 2018.

[74] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[75] Rohan Anil, Vineet Gupta, Tomer Koren, and Yoram Singer. Memory efficient adaptive optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[76] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes, 2020.

[77] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[78] Hanxiao Liu, Andrew Brock, Karen Simonyan, and Quoc V. Le. Evolving normalization-activation layers, 2020.

[79] Zineng Tang, Jaemin Cho, Yixin Nie, and Mohit Bansal. Tvlt: Textless vision-language transformer, 2022.

[80] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[81] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280, December 2020.

[82] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016.

[83] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge?, 2018.

[84] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth, 2016.

[85] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan, 2020.

[86] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks, 2016.

[87] Salam Al-E'mari Yousef Sanajalwe, Mohammed Anbar. Covid-19 automatic detection using deep learning. *Computer Systems Science and Engineering*, 39(1):15–35, 2021.

[88] Mohammad Khalid Pandit, Shoaib Amin Banday, Roohie Naaz, and Mohammad Ahsan Chishti. Automatic detection of covid-19 from chest radiographs using deep learning. *Radiography*, 27(2):483–489, 2021.

[89] Ali Narin, Ceren Kaya, and Ziynet Pamuk. Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks. *Pattern Analysis and Applications*, pages 1–14, 2021.

[90] Fei Zhou, Ting Yu, Ronghui Du, Guohui Fan, Ying Liu, Zhibo Liu, Jie Xiang, Yeming Wang, Bin Song, Xiaoying Gu, et al. Clinical course and risk factors for mortality of adult inpatients with covid-19 in wuhan, china: a retrospective cohort study. *The lancet*, 395(10229):1054–1062, 2020.

[91] Sara Hosseinzadeh Kassania, Peyman Hosseinzadeh Kassanib, Michal J Wesolowskic, Kevin A Schneidera, and Ralph Detersa. Automatic detection of coronavirus disease (covid-19) in x-ray and ct images: a machine learning based approach. *Biocybernetics and Biomedical Engineering*, 41(3):867–879, 2021.

[92] J.C. Ye Y. Oh, S. Park. Deep learning covid-19 features on cxr using limited training data sets. *IEEE Trans. Med. Imaging*, 39(8), 2020.

[93] Zhe et al Xu. Pathological findings of covid-19 associated with acute respiratory distress syndrome. *The Lancet. Respiratory medicine*, 8(4), 2020.

[94] Zhiwen Xiao, Haoxi Zhang, Huagang Tong, and Xin Xu. An efficient temporal network with dual self-distillation for electroencephalography signal classification. In *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1759–1762, 2022.

[95] Sudipta Roy, Tanushree Meena, and Se-Jung Lim. Demystifying supervised learning in healthcare 4.0: A new reality of transforming diagnostic medicine. *Diagnostics*, 12(10), 2022.

[96] Zhiwen Xiao, Xin Xu, Huanlai Xing, Shouxi Luo, Penglin Dai, and Dawei Zhan. Rtfn: A robust temporal feature network for time series classification. *Inf. Sci.*, 571(C):65–86, sep 2021.

[97] Ahmed Al-Karawi et al. Stacked cross validation with deep features: a hybrid method for skin cancer detection. *Tehnički glasnik*, 16(1):33–39, 2022.

[98] Ercan Avşar. Effects of image preprocessing on the performance of convolutional neural networks for pneumonia detection. In *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–5, 2021.

[99] Anwesh Kabiraj, Tanushree Meena, Pailla Balakrishna Reddy, and Sudipta Roy. Detection and classification of lung disease using deep learning architecture from x-ray images. In *Advances in Visual Computing: 17th International Symposium, ISVC 2022, San Diego, CA, USA, October 3–5, 2022, Proceedings, Part I*, page 444–455, Berlin, Heidelberg, 2022. Springer-Verlag.

[100] Huanlai Xing, Zhiwen Xiao, Rong Qu, Zonghai Zhu, and Bowen Zhao. An efficient federated distillation learning system for multitask time series classification. *IEEE Transactions on Instrumentation and Measurement*, 71:1–12, 2022.

[101] Gianluca Maguolo and Loris Nanni. A critic evaluation of methods for covid-19 automatic detection from x-ray images. *Information Fusion*, 76:1–7, 2021.

[102] Navid Razmjooy, Saied Razmjooy, Zahra Vahedi, Vania Estrela, and Gabriel Gomes de Oliveira. Skin color segmentation based on artificial neural network improved by a modified grasshopper optimization algorithm. *Lecture Notes in Electrical Engineering*, pages 169–185, 11 2020.

[103] Qing Liu, Zhigang Liu, Shenghui Yong, Kun Jia, and Navid Razmjooy. Computer-aided breast cancer diagnosis based on image segmentation and interval analysis. *Automatika*, 61(3):496–506, 2020.

[104] Ramin Ranjbarzadeh, Shadi Dorosti, Saeid Jafarzadeh Ghoushchi, Sadaf Safavi, Navid Razmjooy, Nazanin Tataei Sarshar, Shokofeh Anari, and Malika Bendechache. Nerve optic segmentation in ct images using a deep learning model and a texture descriptor. *Complex and Intelligent Systems*, 8(4):3543–3557, Aug 2022.

[105] Debojyoti Pal, Pailla Balakrishna Reddy, and Sudipta Roy. Attention uw-net: A fully connected model for automatic segmentation and annotation of chest x-ray. *Computers in Biology and Medicine*, 150:106083, 2022.

[106] Venkatesan Rajinikanth, J Gnanasoundharam, Seifedine Kadry, Mazin Mohammed, and G Devadhas. Unet with two-fold training for effective segmentation of lung section in chest x-ray. In *2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT)*, pages 977–981. IEEE, 10 2022.

[107] Biraja Ghoshal and Allan Tucker. Estimating uncertainty and interpretability in deep learning for coronavirus (covid-19) detection. *arXiv*, 2020.

[108] Prabira Kumar Sethy and Santi Kumari Behera. Detection of coronavirus disease (covid-19) based on deep features. *electrical and electronic engineering*, 2020.

[109] Tianyang Li, Zhongyi Han, Benzheng Wei, Yuanjie Zheng, Yanfei Hong, and Jinyu Cong. Robust screening of covid-19 from chest x-ray via discriminative cost-sensitive learning. *arXiv preprint arXiv:2004.12592*, 2020.

[110] Jianpeng Zhang, Yutong Xie, Yi Li, Chunhua Shen, and Yong Xia. Covid-19 screening on chest x-ray images using deep learning based anomaly detection. *arXiv preprint arXiv:2003.12338*, 27:141, 2020.

[111] Dimas Reynaldi D. S, Benny Sukma Negara, Suwanto Sanjaya, and Eki Satria. Covid-19 classification for chest x-ray images using deep learning and resnet-101. In *2021 International Congress of Advanced Technology and Engineering (ICOTEN)*, pages 1–4, 2021.

[112] Mazin Mohammed, Belal Al-Khateeb, Mohammed Al-Yousif, Salama Mostafa, Seifedine Kadry, Karrar Abdulkareem, and Begoña Zapirain. Novel crow swarm optimization algorithm and selection approach for optimal deep learning covid-19 diagnostic model. *Computational Intelligence and Neuroscience*, 2022:1–22, 08 2022.

[113] Lawrence O Hall, Rahul Paul, Dmitry B Goldgof, and Gregory M Goldgof. Finding covid-19 from chest x-rays using deep learning on a small dataset. *arXiv preprint arXiv:2004.02060*, 2020.

[114] Mohammad Rahimzadeh and Abolfazl Attar. A new modified deep convolutional neural network for detecting covid-19 from x-ray images. *Molecular Biology*, 04 2020.

[115] Eduardo Luz, Pedro Silva, Rodrigo Silva, et al. Towards an effective and efficient deep learning model for covid-19 patterns detection in x-ray images. *Research on Biomedical Engineering*, page 149–162, 2021.

[116] Yifan Zhang, Shuaicheng Niu, Zhen Qiu, Ying Wei, Peilin Zhao, Jianhua Yao, Junzhou Huang, Qingyao Wu, and Mingkui Tan. Covid-da: Deep domain adaptation from typical pneumonia to covid-19. *arXiv preprint arXiv:2005.01577*, 2020.

[117] Dailin Lv, Wuteng Qi, Yunxiang Li, Lingling Sun, and Yaqi Wang. A cascade network for detecting covid-19 using chest x-rays. *arXiv*, 2020.

[118] Xingwu Zhang, Rui Ma, Ming Li, Xiaolong Li, Zhibo Yang, Ruqiang Yan, and Xuefeng Chen. Feature enhancement based on regular sparse model for planetary gearbox fault diagnosis. *IEEE Transactions on Instrumentation and Measurement*, 71:1–16, 2022.

[119] Yujin Oh, Sangjoon Park, and Jong Chul Ye. Deep learning covid-19 features on cxr using limited training data sets. *IEEE transactions on medical imaging*, 39(8):2688–2700, 2020.

[120] Narinder Singh Punn and Sonali Agarwal. Automated diagnosis of covid-19 with limited posteroanterior chest x-ray images using fine-tuned deep neural networks. *Applied Intelligence*, 51(5):2689–2702, 2021.

[121] Boyi Liu, Bingjie Yan, Yize Zhou, Yifan Yang, and Yixian Zhang. Experiments of federated learning for covid-19 chest x-ray images. *arXiv*, 2020.

[122] Soumick Chatterjee, Fatima Saad, Chompunuch Sarasaen, Suhita Ghosh, Valerie Krug, Rupali Khatun, Rahul Mishra, Nirja Desai, Petia Radeva, Georg Rose, et al. Exploration of interpretability techniques for deep covid-19 classification using chest x-ray images. *arXiv*, 2022.

[123] Laith Mohammad Qasim Abualigah. *Conclusion and Future Work*, pages 163–165. Springer International Publishing, Cham, 2019.

[124] Jeffrey O Agushaka, Absalom E Ezugwu, and Laith Abualigah. Dwarf mongoose optimization algorithm. *Computer methods in applied mechanics and engineering*, 391:114570, 2022.

[125] Laith Abualigah, Dalia Yousri, Mohamed Abd Elaziz, Ahmed A Ewees, Mohammed AA Al-Qaness, and Amir H Gandomi. Aquila optimizer: a novel meta-heuristic optimization algorithm. *Computers & Industrial Engineering*, 157:107250, 2021.

[126] Laith Abualigah, Mohamed Abd Elaziz, Putra Sumari, Zong Woo Geem, and Amir H Gandomi. Reptile search algorithm (rsa): A nature-inspired meta-heuristic optimizer. *Expert Systems with Applications*, 191:116158, 2022.

[127] Olaide Nathaniel Oyelade, Absalom El-Shamir Ezugwu, Tehnan IA Mohamed, and Laith Abualigah. Ebola optimization search algorithm: A new nature-inspired metaheuristic optimization algorithm. *IEEE Access*, 10:16150–16177, 2022.

[128] Laith Abualigah, Ali Diabat, Seyedali Mirjalili, Mohamed Abd Elaziz, and Amir H Gandomi. The arithmetic optimization algorithm. *Computer methods in applied mechanics and engineering*, 376:113609, 2021.

[129] Muhammad Farooq and Abdul Hafeez. Covid-resnet: A deep learning framework for screening of covid19 from radiographs. *arXiv*, 2020.

[130] Nour Eldeen M Khalifa, Mohamed Hamed N Taha, Aboul Ella Hassanien, and Sally Elghamrawy. Detection of coronavirus (covid-19) associated pneumonia based on generative adversarial networks and a fine-tuned deep transfer learning model using chest x-ray dataset. In *Proceedings of the 8th International Conference on Advanced Intelligent Systems and Informatics 2022*, pages 234–247. Springer, 2022.

[131] Siham Tabik, Anabel Gómez-Ríos, José Luis Martín-Rodríguez, Iván Sevillano-García, Manuel Rey-Area, David Charte, Emilio Guirado, Juan-Luis Suárez, Julián Luengo, MA Valero-González, et al. Covidgr dataset and covid-sdnet methodology for predicting covid-19 based on chest x-ray images. *IEEE journal of biomedical and health informatics*, 24(12):3595–3605, 2020.

[132] Ezz El-Din Hemdan, Marwa A Shouman, and Mohamed Esmail Karar. Covidx-net: A framework of deep learning classifiers to diagnose covid-19 in x-ray images. *arXiv*, 2020.

[133] Ankita Shelke, Madhura Inamdar, Vruddhi Shah, Amanshu Tiwari, Aafiya Hussain, Talha Chafekar, and Ninad Mehendale. Chest x-ray classification using deep learning for automated covid-19 screening. *SN computer science*, 2(4):300, 2021.

[134] Shervin Minaee, Rahele Kafieh, Milan Sonka, Shakib Yazdani, and Ghazaleh Jamalipour Soufi. Deep-covid: Predicting covid-19 from chest x-ray images using deep transfer learning. *Medical image analysis*, 65:101794, 2020.

[135] Mete Ahishali, Aysen Degerli, Mehmet Yamac, Serkan Kiranyaz, Muhammad EH Chowdhury, Khalid Hameed, Tahir Hamid, Rashid Mazhar, and Moncef Gabbouj. Advance warning methodologies for covid-19 using chest x-ray images. *arxiv*, abs/2006.05332, 2020.

[136] Mohamed Samir Boudrioua. Covid-19 detection from chest x-ray images using cnns models: Further evidence from deep transfer learning. *The University of Louisville Journal of Respiratory Infections*, 4(1), 2020.

[137] Dalia Ezzat, Aboul Ella Hassanien, and Hassan Aboul Ella. An optimized deep learning architecture for the diagnosis of COVID-19 disease based on gravitational search optimization. *Applied Soft Computing*, 98:106742, jan 2021.

[138] Chun-Fu Yeh, Hsien-Tzu Cheng, et al. A cascaded learning strategy for robust covid-19 pneumonia chest x-ray screening. *arXiv*, 2020.

[139] Sivaramakrishnan Rajaraman, Jenifer Siegelman, Philip O Alderson, Lucas S Folio, Les R Folio, and Sameer K Antani. Iteratively pruned deep learning ensembles for covid-19 detection in chest x-rays. *Ieee Access*, 8:115041–115050, 2020.

[140] Xin Li, Chengyin Li, and Dongxiao Zhu. Covid-mobilexpert: On-device covid-19 patient triage and follow-up using chest x-rays. *arxiv*, 2020.

[141] Luca Brunese, Francesco Mercaldo, Alfonso Reginelli, and Antonella Santone. Explainable deep learning for pulmonary disease and coronavirus covid-19 detection from x-rays. *Computer Methods and Programs in Biomedicine*, 196:105608, 2020.

[142] Morteza Heidari, Seyedehnafiseh Mirniaharikandehei, Abolfazl Zargari Khuzani, Gopichandh Danala, Yuchen Qiu, and Bin Zheng. Improving the performance of cnn

to predict the likelihood of covid-19 using chest x-ray images with preprocessing algorithms. *International journal of medical informatics*, 144:104284, 2020.

[143] Ioannis D Apostolopoulos and Tzani A Mpesiana. Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks. *Physical and engineering sciences in medicine*, 43:635–640, 2020.

[144] Nallamothu Sri Kavya, Thotapalli shilpa, N. Veeranjaneyulu, and D. Divya Priya. Detecting covid19 and pneumonia from chest x-ray images using deep convolutional neural networks. *Materials Today: Proceedings*, 64:737–743, 2022.

[145] Asmaa Abbas, Mohammed M Abdelsamea, and Mohamed Medhat Gaber. Classification of covid-19 in chest x-ray images using detrac deep convolutional neural network. *Applied Intelligence*, 51:854–864, 2021.

[146] Asif Iqbal Khan, Junaid Latief Shah, and Mohammad Mudasir Bhat. Coronet: A deep neural network for detection and diagnosis of covid-19 from chest x-ray images. *Computer methods and programs in biomedicine*, 196:105581, 2020.

[147] Krishna Kant Singh, Manu Siddhartha, and Akansha Singh. Diagnosis of coronavirus disease (covid-19) from chest x-ray images using modified xceptionnet. *Romanian Journal of Information Science and Technology*, 23(657):91–115, 2020.

[148] Halgurd S Maghdid, Aras T Asaad, Kayhan Zrar Ghafoor, Ali Safaa Sadiq, Seyedali Mirjalili, and Muhammad Khurram Khan. Diagnosing covid-19 pneumonia from x-ray and ct images using deep learning and transfer learning algorithms. In *Multimodal image exploitation and learning 2021*, volume 11734, pages 99–110. SPIE, 2021.

[149] S. Q. Salih, H. K. Abdulla, Z. S. Ahmed, N. M. S. Surameery, and R. D Rashid. Modified alexnet convolution neural network for covid-19 detection using chest x-ray images. *Kurdistan Journal of Applied Research*, 11734:119–130, 2020.

[150] Jamal N. Hasoon, Ali Hussein Fadel, Rasha Subhi Hameed, Salama A. Mostafa, Bashar Ahmed Khalaf, Mazin Abed Mohammed, and Jan Nedoma. Covid-19 anomaly detection and classification method based on supervised machine learning of chest x-ray images. *Results in Physics*, 31:105045, 2021.

[151] Unsa Maheen, Khawar Iqbal Malik, and Gohar Ali. Comparative analysis of deep learning algorithms for classification of covid-19 x-ray images, 2021.

[152] Vishal Shenoy and Sachin B. Malik. Covxr: Automated detection of covid-19 pneumonia in chest x-rays through machine learning, 2021.

[153] Julianna Antonchuk, Benjamin Prescott, Philip Melanchthon, and Robin Singh. Covid-19 pneumonia and influenza pneumonia detection using convolutional neural networks, 2021.

[154] Jingyao Liu, Wanchun Sun, Xuehua Zhao, Jiashi Zhao, and Zhengang Jiang. Deep feature fusion classification network (dffcnet): Towards accurate diagnosis of covid-19 using chest x-rays images. *Biomedical Signal Processing and Control*, 76:103677, 2022.

[155] Venkatesan Rajinikanth, J Gnanasoundharam, Seifedine Kadry, Mazin Mohammed, and G Devadhas. Unet with two-fold training for effective segmentation of lung section in chest x-ray. In *2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT)*, pages 977–981. IEEE, 10 2022.

[156] Safa Ben Atitallah, Maha Driss, Wadii Boulila, and Henda Ben Ghezala. Randomly initialized convolutional neural network for the recognition of covid-19 using x-ray images. *Int J Imaging Syst Technol.*, 32:55–73, 05 2022.

[157] Muhab Hariri and Ercan Avşar. Covid-19 and pneumonia diagnosis from chest x-ray images using convolutional neural networks. *Network Modeling Analysis in Health Informatics and Bioinformatics*, 12(1):17, Mar 2023.

[158] Linda Wang, Zhong Qiu Lin, and Alexander Wong. Covid-net: a tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. *Scientific Reports*, 10(1):19549, Nov 2020.

[159] Andy Zhao, Alexander Wong Hossein, Hayden Gunraj, Naomi Waterloo, and May Pal. covidx cxr2, 2020. Online.

[160] Wei Hao Khoong. covid19-xray dataset (train test sets), 2020. Online.

[161] Nabeel Sajid. covid 19-x-ray 10000 images, 2020. Online.

[162] Prashant Patel. chest xray covid19 pneumonia, 2021. Online.

[163] Alif Rahman. covid19 chest xray image dataset, 2021. Online.

[164] praveen govi. coronahack chest xray dataset, 2020. Online.

[165] Zabirul Islam Amanullah Asraf. covid19 pneumonia normal chest xray pa dataset, 2021. Online.

[166] Andrew M V Dadario. Covid-19 x rays, 2020. Online.

[167] Chaudhary Yash. Covid-gan and covid-net mini chest x-ray, 2020. Online.

[168] Daniel Kermany, Kang Zhang, and Michae Goldbaum. Labeled optical coherence tomography (oct) and chest x-ray images for classification. 2018.

[169] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

[170] ITU-R. Bt.709 : Parameter values for the hdtv standards for production and international programme exchange. 2015.

[171] ITU-R. Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios. 2007.

[172] Nicolas Limare, Jose-Luis Lisani, Jean-Michel Morel, Ana-Belen Petro, and Catalina Sbert. simplest color balance. *Image Processing On Line*, 1:297–315, 10 2011.

[173] J. A. Stephen Viggiano et al. Comparison of the accuracy of different white-balancing options as quantified by their color constancy. *Proc. SPIE 5301, Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications*, 2004.

[174] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 9351:234–241, 2015.

[175] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, Los Alamitos, CA, USA, jun 2015. IEEE Computer Society.

[176] Stefan Jaeger, Sema Candemir, Sameer Antani, Yì-Xiáng J. Wáng, Pu-Xuan Lu, and George Thoma. Two public chest x-ray datasets for computer-aided screening of pulmonary diseases. *Quantitative Imaging in Medicine and Surgery*, 4(6), 2014.

[177] Muhammad Enamul Hoque Chowdhury, Tawsifur Rahman, Amith Khandakar, Rashid Mazhar, Muhammad Abdul Kadir, Zaid Bin Mahbub, Khandakar Reajul Islam, Muhammad Salman Khan, Atif Iqbal, Nasser Al-Emadi, and Mamun Bin Ibne Reaz. Can AI help in screening viral and COVID-19 pneumonia? *CoRR*, abs/2003.13145, 2020.

[178] Società Italiana di Radiologia Medica e Interventistica. Covid-19 database, 2020. online.

[179] Maria Iglesia Vayá, Jose Manuel Saborit-Torres, Joaquim Angel Montell Serrano, Elena Oliver-Garcia, Antonio Pertusa, Aurelia Bustos, Miguel Cazorla, Joaquin Galant, Xavier Barber, Domingo Orozco-Beltrán, Francisco García-García, Marisa Caparrós, Germán González, and Jose María Salinas. Bimcv covid-19+: a large annotated dataset of rx and ct images from covid-19 patients, 2021.

[180] Anouk Stein, MD, Carol Wu, Chris Carr, George Shih, Jamie Dulkowski, kalpathy, Leon Chen, Luciano Prevedello, Marc Kohli, Mark McDonald, Peter, Phil Culliton, Safwan Halabi, and Tian Xia. Rsna pneumonia detection challenge, 2018.