LUISS

Degree Program in Data Science and Management

Course of Machine Learning

Machine Learning in the Healthcare Sector: The Development of a Pneumonia Segmentation Model

Prof. Giuseppe Francesco Italiano

SUPERVISOR

Prof. Antonio Simeone

CO-SUPERVISOR

Leonardo Berrettoni - 759321

CANDIDATE

Academic Year 2023/2024

Sommario

1. INTRODUCTION

1.1. General Introduction

Machine learning is one of the most discussed topics of our days thanks to his capability to reshape industries and revolutionize how data is utilized. Since Alan Turing and John McCarthy first discussed the topics, the subject has had an incredible journey of advancement; machine learning (ML) and artificial intelligence (AI), from their beginnings to their current broad deployment across a wide range of industries, have fundamentally changed our understanding of technology and its potential (Eswaran & Khang, 2024). Every year, advances in algorithms, processing power, and data accessibility push the envelope of what is conceivable, pointing to a day, closer than someone might imagine, when intelligent systems will be seamlessly incorporated into our daily lives, transforming everything from entertainment and transportation to healthcare and finance.

While John McCarthy was the one proposing for the first time the term “artificial intelligence” at a Dartmouth conference, Alan Turing have been a pioneer on the applied machine learning working and developing during World War II the ‘Enigma Machine’, capable of breaking the German ciphers (Eswaran & Khang, 2024).

Turing’s Machine was able to perform mathematical calculations automatically and thanks to this, he laid the groundwork for the development of artificial intelligence (AI) as we know it today. AI and ML based tools are designed to carry out intelligent tasks that are typically completed by people, evaluating and learning from data using specially created algorithms.

1.2. State of Art: ML in the Healthcare Sector

Since artificial intelligence (AI) was established as a field of study in the 1950s, a great deal of research has been done in areas such natural language processing, learning, reasoning, and knowledge representation. As a result, artificial intelligence has been used in many fields, including marketing, games, robotics, e-commerce, healthcare, agriculture, and education and more specifically, search engines like Google, recommender systems like Netflix, self-driving cars like Tesla, and voice recognition

software like Siri and Alexa are examples of commonly used AI applications (O. Ali et al., 2023).

Beside all the possible fields where ML could be flourishingly applied one of the most promising is the healthcare sector, offering new ways to enhance diagnostic accuracy, improving patient outcomes and reducing costs, providing ongoing support to operators based on previous analyses in order to reduce the % of wrong diagnosis. In this context, machine learning is not merely a technological innovation but a revolutionary force that is reshaping the way healthcare services are delivered, managed, and optimized (Siddique and Chow, 2021).

Its application in healthcare addresses various challenges from early diagnosis and predictive analytics to personalized medicine and advanced imaging techniques. At its core, machine learning enhances the ability of healthcare providers to diagnose diseases early and accurately, predict patient outcomes, and customize treatment plans based on individual patient data. This level of precision and personalization represents a significant departure from the traditional, one-size-fits-all approach to healthcare, where treatments are typically based on broad population averages rather than tailored to the specific genetic, environmental, and lifestyle factors that influence each patient's health (Javaid et al., 2022).

Early diagnosis and predictive analytics are among the most profitable applications of how machine learning is used in healthcare today. Early disease detection, particularly for conditions with fast progression or those that are difficult to diagnose through conventional means, can significantly improve patient outcomes. Machine learning models trained on large datasets can analyze complex patterns in patient data, such as demographic information, clinical history, lab test results, and imaging data, to predict the onset of diseases before symptoms even appear with greater accuracy. This predictive capability is especially critical in managing chronic diseases like cancer, cardiovascular diseases, and neurological disorders, where early intervention can drastically alter the course of the disease and improve survival rates (Siddique and Chow, 2021).

As mentioned before, a stunning benefit of using ML in the medical area is personalized medicine. This approach, unlike conventional ones that apply standard treatment protocols to all patients, personalized medicine aims to tailor treatments to the unique genetic, environmental, and lifestyle factors of each patient. Machine learning

algorithms play a crucial role in this approach by analyzing large datasets of patient information, including genetic sequences, biomarker profiles, and electronic health records (EHRs), to identify the most effective treatment strategies for individual patients (Panesar, 2019). In oncology, for example, starting from a patient's genetic profile, Machine Learning models have been used to predict a patient's response to chemotherapy enhancing treatment outcomes and reducing the incidence of adverse reactions. Similarly, in psychiatry, Machine Learning algorithms have been employed to predict responses to antidepressants, allowing clinicians to select the most effective medication from the outset, thereby improving patient outcomes and reducing the trial-and-error approach that currently dominates psychiatric treatment (Shailaja et al., 2018).

The use of AI in the healthcare sector is particularly effective in those country that are commonly known as low- and middle-income countries (LMICs), strengthening the whole system and increasing quantity and quality of medical care providing better evaluation and treatments to people who live there (Ciecierski-Holmes et al., 2022). These countries are encountering different types of problems when talking about medical care from qualified Human Resources to availability of field-relevant diagnostics and ML could help in three different areas: (a) clinical decision support at both health center and community levels, (b) population health, and (c) direct patient support. One solution to the shortage of personnel that has already been in place for years is task-shifting, where non-specialists perform the tasks of specialists through protocolized medicine (Williams et al., 2021). Task-shifting has been employed from surgery to dermatology and recently it has started to be backed by AI-enabled tools that have the potential to allow the quality and safety of task-shifting to be made better supported giving another point of view to physicians that are not specialized in that precise field.

The real strength of adopting ML based tools is that they can be installed on smartphones or other medical devices. For the past decade smartphones have played an important role in LMIC healthcare delivery hosting important decision-support tools for family planning, prenatal and antenatal care, intrapartum progression of labor, diagnosis of pediatric pneumonia and many others. The combination of user interface, communications capabilities, audio and video sensors, and local computational capabilities makes modern smartphones able to capture high quality images, video or

audio that can provide an immediate feedback about skin lesions or other disease detectable with these tools (Williams et al., 2021); moreover equipping portable X-Ray scans with ML models can instantly tell you whether you have a broken bone or whether you are suffering from pneumonia.

1.2.1. ML in Radiology

The most significant impacts of machine learning in healthcare is in the field of medical imaging and radiology and as a proof of this, in 2022, 391 tools on 521 authorized by the food and drugs administration concerned exactly radiology (Gaire, 2023). Traditional diagnostic methods, while effective, rely heavily on the expertise of radiologists, who are required to interpret complex medical images such as X-rays, CT scans, and MRIs. Although skilled, especially under conditions of high workload or fatigue, radiologists can miss subtle signs of disease (Tsai and Tao, 2019). As a result, this technology gives doctors comprehensive insights into the medical needs of their patients in addition to gathering and processing patient data. Amisha et al. (2019) points out a 2016 study where it has been found that physicians spend 51% of their workday at their desks and getting electronic health information, with only 27% of their office hours spent in direct patient interaction, they worked on EHR-related chores for more than half of their time in the examination room.

Artificial intelligence (AI) in medicine increases accuracy, productivity, and efficiency while freeing up more time for primary care physicians by reducing the need for manual work. As of now Machine learning-enabled automated systems are adopted as a support to radiologists delivering fast and unbiased evaluations, which is vital considering the millions of annual deaths attributable to pneumonia (Szepesi & Szilágyi, 2022) providing amazing benefits in situations where time is crucial prioritizing cases that require immediate attention (Kareem et al., 2022). They are particularly effective in cases of emergency-urgency, those situations where the emergency room of the first aid area of an hospital is overcrowded and there are no radiologist available because they are

all work loaded; using ML tools to interpret X-Ray scans or any electronical record could speed up the whole diagnostic process (Rajpurkar & Lungren, 2023).

1.3. The Models Used

For managing and processing these images, particularly deep learning models like Convolutional Neural Networks (CNNs), have been shown to enhance the accuracy of medical image interpretation, often outperforming human experts in detecting early signs of diseases like cancer and pneumonia. These models are trained on millions of labeled images, learning to identify disease markers that may be invisible to the human eye, thus improving diagnostic accuracy and enabling earlier interventions. (Reshan et al., 2023).

Convolutional Neural networks are a type of deep learning algorithms that are strongly similar to ANN that try to emulate the human vision system. They have the power to transform images into number and process them identifying what are the critical features that classify an image as belonging to a class or to another. The topic of understanding how to deal with visual input in machine learning has been discussed since 1950s and a milestone has been put in 2012 by the university of Toronto with the development of AlexNet; this model won the ImageNet competition with an 85% accuracy, it *was the first time a CNN was developed* (Mandal, 2024).

From that moment onward, CNN became pivotal in processing and segmenting images, detecting objects, and reducing the images to a more workable format that will give reliable predictions.

Although CNNs are based along the lines of ANN, their structure is quite different:

- The input of CNN is the image as such, depending on its color scale, Black and white or colored, it is translated, respectively, into one or three matrices (one for each RGB color)
- Convolutional Layer: The matrices are then passed to the first layer of CNN, the convolutional layer. The scope of this layer is the one of identifying the most

relevant features of the image such as lines, edges, angles and different shapes. A variables number of small matrices called filters or kernel are passed over the input matrices giving as output another matrix whose dimension is usually smaller than the input one. To be noted that the number of output matrices is equal to the number of filters applied to the input image. On the feature map resulting from this layer are then applied some activation functions. An activation function defines whether to activate a neuron by using the weighted sum of inputs together with a bias term, its main purpose is to put non-linearity into the neural network, enabling it to learn and represent any complicated, arbitrary kind of functions that linear models can't. Non-linear activation functions allow a network to model complex interactions within data and therefore to make better decisions. They also play a very significant role in backpropagation, as it is through the application of such functions that gradient-based learning could effectively update weights. Without them, the network would only learn linear relationships.

The most common functions are:

- Sigmoid (Logistic) Function: $\sigma(x)= \frac{1}{1+e^{-x}}$

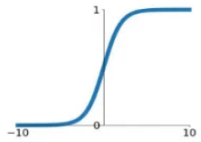


Image 1.4.1 (Jadon, 2022)

- Hyperbolic Tangent (Tanh) Function: $\tanh(x)=\frac{2}{1+e^{-2x}} - 1$

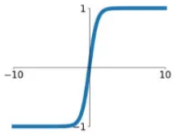


Image 1.4.2 (Jadon, 2022)

- Rectified Linear Unit (ReLU): max(0,x)

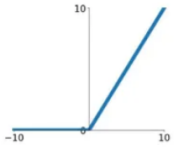


Image 1.4.3(Jadon, 2022)

- Pooling layer: this layer, as the convolutional one, has the role to reduce the dimension of the matrices it takes as input. There exist two different types of pooling layer, a max pooling and average pooling; they both take a fixed-shape subset of the input matrix and take respectively the max value and the average value in the subset and populate with this the resulting feature map.
- Flattening and fully connected layer: the last architectural component of CNN are the flattening layer and the fully connected layer. As of now we have been working with matrices but the input the fully connected layer is expecting to receive has not 2 dimensions but just one. The flattening layer has the role to reshape the last pooled feature map from a 2-dimensional array to a 1-dimensional array. The fully connected layer is the last block of the CNN structure and it resembles a traditional artificial neural network (ANN), where multiple neuron layers are connected until the final layer performs classification based on the probability of each record to belong to one class or another. What makes this block unique is that every neuron in each layer is connected to all neurons in both the preceding and subsequent layers.
- Overfitting avoidance layers: One of the biggest risks when training a broadly designed CNN is the one of falling into overfitting; to avoid this there are some special layers whose role is the one of applying regularization techniques to decrease this type of risk. The two most known are the dropout layer and the normalization layer.

    - The dropout layer: with the terms 'dropout' we refer to the process of deactivating nodes in a neural network, temporarily. When a node is dropped, all forward and backward connections that are associated with the node are removed for the time being. That will result in a modified network architecture derived from the original network. Dropping is done

in a predetermined manner dictated by a dropout probability, generally denoted by p (Yadav, 2023).

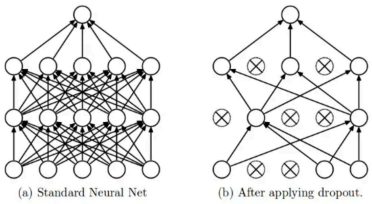


Image 1.4.4 (Yadav, 2023)

- The batch normalization layer: the batch normalization is a layer as all the other and it is particularly useful when dealing with features having very different scales, it normalizes the values of features to avoid that large feature will drown out the small feature. It is called batch normalization because the normalization activity is performed batch by batch when data are passed through the network (Doshi, 2022).

1.4. The Aim of the Research

VGG-16, VGG-19, ResNet50, they are all models applied in radiology, to help physicians in the evaluation process; every developer has his own idea about which one of them is the best one in fulfilling medical tasks. The aim of this project is to build, train and evaluate some of the in order to understand which of them is actually the best one in a well-defined case. The final task that the models must solve in this project is pneumonia detection: understanding, starting from X-Ray images, which patients are suffering from pneumonia and which one are not. Pneumonia is an infection that affects one or both lungs causing the air sacs, or alveoli, of the lungs to fill up with fluid or pus. Symptoms can range from mild to serious and may include a cough fever, chills, and trouble breathing. Age, your overall health, and what caused the infection are all factors that could determine the severity of the illness (What Is Pneumonia? | NHLBI, NIH, 2022).

The models that are going to be compared are ResNet50, VGG16 and an Optimized CNN; the first two are models commonly used for this type of task (Reshan et al., 2023), while the optimized CN has been chosen because CNN is the base of the previously mentioned two and it is the ML architecture used when dealing with images.

ResNet50 and VGG16 are pretrained models, this means that the training process has been already undertaken (the well-known dataset "ImageNet" in this case) allowing each of them to put in place transfer learning. In transfer learning, instead of training a model from scratch, it is leveraged the knowledge already learnt from a general task to improve performance and reduce training time for a related scope. This fine-tunes the model by changing the architecture of the final layer and keeping the other layers of the pre-trained model fixed. In computer vision, this method has proved outstanding performances increasing accuracy and speed. Despite the benefits, this time a usual application of transfer learning was not enough since the ImageNet dataset contains images belonging to several categories but not to the class of chest X-Ray; some other techniques needed to be undertaken like freezing and unfreezing the last layers of the pretrained models in order to have better result, this will be explained more in detail at a later time during the project (Vishal, 2024).

ResNet is short for Residual Networks, is quite famous for the use of skip connections (residual connections), a technique that guarantee the free flow of information and challenge the problem of vanishing gradients in a very deep network. The model sued for solving the set task, is ResNet-50 which consists of 50 layers. The skip connections are known to allow networks to learn residual mappings, which helps a deep network to be trained without the risk of vanishing gradients (Rehman, 2024). VGG16, instead, is a simpler architecture with respect to ResNet50 but it is still very effective. It consists of 16 weighted layers, out of which 13 are convolutional and 3 fully connected. The core of VGG16 is to consistently use small 3x3 size filters of convolutional layers followed by another layer of max pooling where fine image details can be exploited and complex patterns learned (Great Learning, 2022b) (Vepuri, 2022). Both models will be explained more in depth in the next paragraphs.

From this introduction, it could have been probably grasped how can ML be used in the healthcare sector and which are its benefit. Along this project we will firstly go through a literature review highlighting the state of art of this topic pausing even more on its applications, benefits and especially possible drawback; later on a technical explanation of CNN will be undertaken illustrating how they are usually structured and which are their layer and their features. The following sections will be dedicated to a deeper description of the three models mentioned above (ResNet50, VGG16 and the optimized CNN) and their evaluation with the choice of the best model.

2. LITERATURE REVIEW

2.1. Overview

A literature review is defined as an account of the existing literature related to a specific topic or area of interest. The LR investigates how this new auditing tool increases the efficiency of detecting pneumonia using better imaging scans for early diagnosis. It also provides insight into the moral dilemmas when using machine learning in healthcare and some of the issues, like data privacy, bias, and the black box dilemma. This review also highlights the advantages and challenges of integrating machine learning technologies in healthcare.

2.2. Machine Learning in The Healthcare Sector

Machine learning can be described as a set of algorithms and statistical techniques that allows applications to improve their performance and decision-making about some input data based on what has been learnt from other similar data sets without having to be specifically programmed (Habehh and Gohel, 2021). These models are on the level of easy algorithms including linear regression and decision trees up to the complex architectural solutions like deep neural network which is based on many layers of interconnected nodes that imitate the work of the human brain. A major benefit of using the machine learning approach to text classification is the potential to analyze vast datasets and find patterns that a human analyst may not be able to identify (Alanazi, 2022).

In the context of healthcare, machine learning models are usually trained on very large data sets that contain many different types of medical data, including demographic information about the patient such as age, sex, and ethnicity, clinical history of the patient including symptoms and past illnesses, laboratory test results, imaging test results including x-rays and MRI scans and genomic data of the patient (Alanazi, 2022). These models are trained using data to understand the patterns and the dependency between the variables to be able to make predictions about patient status, disease progression, or propensity of certain diseases. Nevertheless, the idea here is that these predictions can be made with reasonable stability and accuracy in comparison with other methods; therefore,

machine learning seems a valuable addition to the variety of approaches applied in the healthcare industry (Callahan and Shah, 2017).

Callahan and Shah (2017) noted that machine learning models can be classified based on the learning methodology. These are supervised learning, unsupervised learning, and reinforcement learning. In Supervised learning the model is trained with the identified set of data which means each data point has its correct output. This kind of training makes the model capable of foreseeing the outcome of a data input that the model has not seen before (Nayyar et al., 2021).

Conversely, the unsupervised learning process is one in which the model is trained on a dataset and there are no labels, in this process, the model is free to find some of the patterns or groupings in the data set by itself (Nayyar et al., 2021).

The third strategy is referred to as the reinforcing learning technique whereby a model is trained to generate a sequence of decisions that should be accompanied by feedback in the form of rewards or penalties based on the outcome. These are learning approaches that possess characteristics of learning that are applicable depending on the nature of the problem under consideration in a system of healthcare (Javaid et al., 2022).

2.2.1. Early Diagnosis and Predictive Analytics

One of the most significant ways through which machine learning has perhaps made the most amount of positive impact in this area is its ability to enhance time to diagnosis and time to prognosis (Javaid et al., 2022). The concept of early disease detection is one of the fundamental tenets of therapeutic activity, as the effectiveness of treatment depends on the stage at which the disease is identified. This is especially the case with diseases with fast-moving or even invisible stages that are not easy to detect through conventional medical means (Siddique and Chow, 2021).

Big data analysis is one of the most significant advantages of machine learning for the identification of disease patterns that could be indicative of a disease's presence even if its symptoms have not emerged yet (Siddique and Chow, 2021). This capability is very helpful in diagnosing cancer, cardiovascular disease, and neurological disorders, where early detection can greatly help the patient's condition (Ahmad et al., 2018).

For example, in the field of oncology, deep-learning models are used to analyze medical images such as mammograms or skin photographs for signs of cancer at an early

stage (Ahmad et al., 2018). These models have been trained using a dataset that may contain hundreds or even thousands of samples with images and labels that enable the model to distinguish between malignant and benign lesions with very high accuracy. At times, such models can perform as effectively as or even better than professional experts in the field, the discovery of which serves as a practical instrument for the early diagnosis of cancer (Shailaja et al., 2018).

For instance deep learning, which is a branch of machine learning, has been used in dermatology as a tool to help diagnose early stages of skin cancer based on images of skin lesions (Shailaja et al., 2018). Such models can diagnose benign and malignant lesions with the highest accuracy, comparable to dermatologists. These models are trained on information images and in this process, learn to differentiate between almost similar features such as color, texture, and shape that might not be easily discernable. This capability not only plays the role of early diagnosis of skin cancer but also increases the minimally invasive biopsies thus increasing the comfort of the patient and decreasing the cost of health care (Saleem and Chishti, 2020).

In addition to cancer detection, early diagnosis uses machine learning ways. It also provides information for the prediction of the occurrence of chronic diseases and other severe health conditions (Saleem and Chishti, 2020). Computer algorithms that can learn from experience can analyze the data on patient's demographics, medical history, genetics, and even the environment to provide predictions about possible upcoming acute health issues (Bhardwaj et al., 2017).

For instance, machine learning has been used in the prediction of chronic diseases such as diabetes, hypertension, and heart diseases among others (Toh and Brody, 2021). It can arrange these people according to the patterns from the EHRs, which the healthcare providers can then use in taking preventive measures for these diseases. This preventive measure does not only reduce the incidence of chronic diseases but also the overall health of the people is improved through early diagnosis and altering behavior (Bhardwaj et al., 2017).

ML help healthcare providers develop subsequent care management strategies for individual users depending on the identified threats to minimize the likelihood of readmissions (Qayyum et al., 2020). For instance, the prediction of a reassessment of a patient with heart failure and other associated diseases may be considered high risk by a

machine learning model. Instead, the healthcare team can offer closer monitoring services, either as regular appointments, changes in medicines, or home healthcare services to avoid being readmitted (Toh and Brody, 2021).

Healthcare operations are also advanced by using predictive analytics based on machine learning to effectively and efficiently allocate resources (Toh and Brody, 2021). For instance, hospitals can employ auto-regression models to forecast the number of admissions depending on history data, seasonality, and local outbreaks. These predictions help healthcare facilities to have a good staff who can help in operations, efficient utilization of resources, and also less exploiting time for the patients. For instance, in flu season, a hospital can hypothesize on the likelihood of a higher number of patient admissions and hence pre-empt by hiring more staff, purchasing more stocks, and arranging for more beds. As a result of achieving operational efficiencies, there is improved patient satisfaction, overall health system capacity is enhanced, and pressures during a surge in patient demand are kept to a minimum (Panesar, 2019).

2.2.2. Personalized Medicine

Precision medicine or, as it is also called, personalized medicine is an innovative model of healthcare delivery that aims to provide a patient-specific treatment (Panesar, 2019). It is also different from the conventional approach of developing a standard treatment plan depending on the disease's type but uses genetic, environmental, and other aspects of a patient as well as biochemical markers in creating an intervention plan that is either more effective, safe, or faster. Machine learning plays a crucial role in the implementation of individualized medicine as it provides high potential for the analysis of large datasets, outcome prediction, and the identification of the most suitable treatment approaches (Zhang et al., 2022).

Another successful use case incorporating machine learning into the concept of personalized medicine is pharmacogenomics, the branch of medicine that aims to identify the connection between the patient's genes and the effectiveness of medication (Zhang et al., 2022). It is possible to extract features from genetic data and use machine learning to forecast the mode of reaction individuals will show to certain drugs, which helps in the development of personalized treatment plans. Apart from enhancing the effectiveness of treatment regimens, this approach minimizes the possibility of adverse drug reactions

whose effects are fathomless in the sphere of health care. For instance, the way a patient metabolizes drugs including warfarin, a popular anticoagulant, depends on inherited genes (Kasula, 2021).

In oncology, for instance, it has been applied in creating an individualized treatment plan for cancer patients (Kasula, 2021). The work done involved the use of gene expression data from the patients to train a SVM, support vector machine, which is a type of machine learning algorithm to predict response to chemotherapy. All the 175 trial participants had cancer and their gene expressions were used to determine whether they would metabolize the chemotherapies commonly administered to cancer patients. The analysis offered quite positive outcomes with the variation in the percentage of predictions higher than 80% for various types of drugs. That is why machine learning can be viewed as one of the key tools for improving the effectiveness of cancer treatment and the determination of the most suitable therapy for patients based on their genetic characteristics (Sabry et al., 2022).

Another area where machine learning is important is in predicting outcomes of psychiatric treatments (Sabry et al., 2022). In a study focused on the identification of the patient's response to various classes of antidepressants, the authors utilized the EHRs of 17,556 individuals and machine learning techniques. From the AI models, features that are predictive of treatment selection have been used to reduce confounding factors and the models gave good predictions. This research showed that, based on real-world EHRs, AI modeling could identify patients who would respond well to antidepressants, which implies that such techniques might be useful for implementing integrated clinical decision support systems to enhance treatment efficiency. It could also mean the development of better-targeted treatments for conditions that are known to require testing many drugs and dosages before finding one that works well (Pattnayak and Panda, 2021).

Although AI techniques and the inclusion of genomics to predict treatment efficacy have seen a lot of advancement, there is a need for more prospective and retrospective clinical studies (Pattnayak and Panda, 2021). These efforts are critical for collecting the massive and diverse data sets that are necessary to successfully train machine learning algorithms, for validating their usefulness and effectiveness within the healthcare setting, and for continuing the advancement of AI-based clinical decision support systems. The further application of these tools for the development of

personalized medicine in healthcare maintains hope for enhancing good results in every stream of medical care for patients (Swain et al., 2022).

2.2.3. Imaging and Radiology

There has been the advancement of machine learning in medical imaging and radiology, which plays an important role in assessing and diagnosing several diseases (Swain et al., 2022). The capabilities of the ML algorithms to classify medical images with high accuracy have dramatically impacted these sectors to deliver accurate diagnoses earlier. Imaging has become a significant part of today's healthcare systems and it entails capturing visual images of the human body which can be used to diagnose a myriad of diseases, for instance, a broken bone, and cancerous tumors among others (Mohanty et al., 2021).

In radiology, machine learning models are employed to make interpretations of images acquired from different modalities such as X-ray, MRI (magnetic resonance imaging), CT (computed tomography), and ultrasound scans (Mohanty et al., 2021). These models are trained with massive data from the medical image database and they are capable of learning the patterns that are linked to a given condition. For instance, breast cancer screening machines were designed to analyze mammograms and identify tumors in earlier stages than any other method. This is important because early-stage cancers are easier to treat than the later stages. Thus, using data on calcifications and other features that may suggest the existence of a tumor, the machine learning algorithm helps radiologists identify carcinomas that can remain unnoticed during ordinary scans (Sanchez et al., 2022).

Likewise, deep learning has demonstrated tremendous potential in the diagnosis of lung cancer from chest X-rays and CT scans (Sanchez et al., 2022). Pulmonary cancer is intrinsically aggressive to diagnose in its early stages since its symptoms resemble those of other common and less severe respiratory diseases, while small-sized carcinomas are disregarded. However, recent studies in the field of machine learning have shown the potential for detecting small and often overlooked symptoms of the disease like nodules in lung tissue. A recent real-world lung cancer screening example identifies early-stage lung cancer using a deep learning model on a large database of chest CT scans with

accuracy that is comparable to that of expert radiologists (Mustafa and Rahimi Azghadi, 2021).

ML is also being applied in the clinical diagnosis of skin disorders through the use of machine learning (Mustafa and Rahimi Azghadi, 2021). For instance, convolutional neural networks which is a subcategory of deep learning have been utilized in the assessment of images of skin lesions and differentiation between skin cancers, specifically melanoma. Such models have shown great effectiveness that is even comparable with the effectiveness of dermatologists, and this means that early diagnostics are possible (Ahmad et al., 2020).

In ophthalmology, machine learning models are being applied in the diagnosis and progress tracking of eye pathologies like DR (diabetic retinopathy) and AMD, age-related macular degeneration (Ahmad et al., 2020). Diabetic retinopathy is one of the most common complications of diabetes that can lead to blindness if left untreated, and timely screening is crucial. Deriving from the earlier point, patients with DR be screened for microaneurysms or hemorrhages using retinal images analyzed by machine learning models hence allowing early corrective action to be taken to prevent the worsening of the DR (Chua et al., 2023).

Likewise, machine learning models assist ophthalmologists in supervising AMD, a leading cause of vision impairment in seniors, by scanning patients' retinal images for indicators of disease development (Chua et al., 2023). These models can assist the ophthalmologist in the decision-making process regarding diagnoses and interventions including anti-VEGF injections that may arrest or reverse the AMD progression (Gupta and Sedamkar, 2020).

In addition, machine learning has been used to improve the quality and effectiveness of medical imaging (Gupta and Sedamkar, 2020). For instance, in scanning and imaging, ML algorithms can fine-tune image reconstruction; this capability is highly useful in MRI and CT imaging because faster and more accurate scans result in faster diagnosing and therefore improved patient care. However, it is also possible for machine learning models to contribute to the interpretation of images that are ambiguous and hard to dissect for a diagnosis, and come up with useful information that would benefit the radiologists (An et al., 2023).

Machine learning has also been used to monitor and bring prognosis for neurodegenerative diseases including Alzheimer's disease, Parkinson's disease, and severe mental illnesses like psychosis depression, and PTSD (An et al., 2023). Alzheimer's disease has no known cure but if diagnosed early, one can start planning and trying to keep the symptoms from appearing or from worsening (Usmani and Jaafar, 2022). For example, machine learning models can review imagery like MRI or CT scans and consider biomarkers that signify initial potential indicators of neurodegenerative diseases, like cortical thinning or hippocampal atrophy, which can then assist with treatment planning (Arvindhan et al., 2021).

Likewise, the prediction of diagnosis and treatment outcomes of depression has been done through machine learning techniques (Arvindhan et al., 2021). In one study, they employed decision tree models and feature-richer datasets including fMRI, CB (cognitive behavior) scores, and age to build a predictive model of depression. On diagnosis, the model had an accuracy of 87.27% while on treatment response it was 89.47%. This predictive capability is particularly useful in mental health where, wherein conditions such as depression, diagnosis, and treatment are greatly hampered by issues such as subjective reporting of symptoms and high variability in responsiveness to treatment (Gichoya et al., 2021).

The present uses of machine learning in medical imaging and radiology are the perfect example of the benefits of technology in the further development of the medical industry (Gichoya et al., 2021). The effectiveness of machine learning in terms of accuracy, classification, sensitivity, and specificity underlines their importance in healthcare systems. Over time and with advancing technology, the application of machine learning in medical imaging will only grow, and its capabilities for enhancing the diagnosis process and improving the health of patients will increase (Chen et al., 2021).

2.2.4. Genetic Engineering and Genomics

Genetic engineering and genomics are some of the areas that have been revolutionized by machine learning (Chen et al., 2021). CRISPR-Cas9 gene has provided a new tool for making modifications within the human genome, where a researcher can effectively alter the DNA sequence of target genes. Nonetheless, the effectiveness of these methods relies on the accuracy of the genetic alterations as random changes can

pose negative repercussions. Machine learning has also emerged as a central way of increasing genetic and other genomic tasks' precision and speed, to guarantee that edits are done correctly and without consequences on other genes (MacKay et al., 2023).

Another important issue regarding manipulations at the DNA level is dealing with the issue of off-target effects, that is, secondary effects that are realized at a site other than the intended one (MacKay et al., 2023). The effects that are produced by the mutation can be random and in some cases can be risky since they introduce abnormality within the protein. To overcome this challenge, new machine learning modes have been designed to forecast where one of these off-target effects might happen allowing enhanced accuracy of the CRISPR-Cas9 gene editing. For instance, in deep learning models, the genome can be categorized to look for possible off-target sites due to high sequence homology to the target site. Thus, such identification will help researchers to create more suitable guide RNAs, which would minimize the likelihood of off-target effects, thus enhancing the safety of the gene editing process (Feng et al., 2022).

Machine learning is also being used for the rational design of enhanced high-fidelity Cas9 variants thereof Cas9 protein that are superior in gene editing (Feng et al., 2022). The high-fidelity variants moreover contain fewer off-target effects, suggesting that they are safer to be utilized in clinical practice. With big data on genomics accessible, machine learning approaches can determine how different Cas9 variants should be utilized for various applications. For instance, by applying machine learning, one could obtain data from several experiments in gene editing outcomes to identify which Cas9 variant is most effective in achieving the required edit frequency with minimal off-target impact. Such information can be used in modifying newer Cas9 variants hence extending the knowledge in genetic engineering (Rasheed et al., 2022).

Besides, Gene editing has also revolutionized the field of genomics through the application of machine learning in genetic big data analysis (Rasheed et al., 2022). Genomic data is highly detailed, containing millions of different data points, which may reflect various characteristics of an individual's genome. Conventional techniques for processing this type of information may take a long time, in addition, a web of interconnection between the various genetic factors may not be fully revealed by traditional approaches. Nevertheless, with the help of machine learning techniques, such data can be processed at a higher speed and can reveal characteristics or dependencies

that can be imperceptible to experts. This capability is the most important for undertaking research, where diseases with multiple genes and environmental effects play a role (Rastogi et al., 2022).

A major use of machine learning in genomics is in the identification of certain diseases that are likely to affect an individual given his or her genotype (Rastogi et al., 2022). Machine learning algorithms can employ algorithms on large population genetic databases and thus predict key genes that are associated with the kinds of diseases including cancer, heart diseases, and diabetes among others. For instance, a machine learning algorithm can employ genotyping data from thousands of people to determine which genetic markers are significantly associated with a particular disease.

Such knowledge can then be utilized in the creation of questionnaires which can help in identifying those individuals who are at a high risk of developing these diseases; this may help in disease prevention. For example, those people with a genetic predisposition to an illness like breast cancer may agree to be subjected to more tests, have prophylactic surgery, or go for chemoprevention as a preventive measure (McCoy et al., 2020).

Machine learning is also valuable in the establishment of the concept of personalized medicine with a specific focus on pharmacogenomics (McCoy et al., 2020). Pharmacogenomics is analyzed as the relationship between genetic factors and the use of drugs. Machine learning models can then appropriately make predictions regarding patients' reactions to certain drugs and adjust therapies to conform to patients' genetic information. It also helps enhance the effectiveness of treatment and minimizes the incidences of adverse drug reactions, which may be serious or fatal. For instance, an ML model might take a patient's genes and predict if the patient will metabolize a specific drug quickly or slowly. Such information can be used to further fine-tune the dosage or even select another product that will be better for a particular patient, thus providing the most effective and safe therapy (Rani et al., 2023).

Machine learning has also contributed to genetic engineering in combating COVID-19 (Rani et al., 2023). Scientists have recently employed software developed through machine learning approaches to identify which antigens possess the characteristics of HLA-binding, processing, presentation to the cell surface, and T-cell recognition—attributes necessary for successful immunotherapy targets (Jia et al., 2022).

With the help of the machine learning algorithm performing the sequence analysis of the SARS-CoV-2 virus, the epitope hotspots were discovered, which means that the regions of the virus modeled by the immune system as dangerous are identified (Jia et al., 2022). This information helped in deciding on the advancements in vaccines and immunotherapies, in the conceptual construct of a universal vaccine, useful for the entire population of the world. Such an approach can be effective and demonstrate the ability to deploy contemporary machine learning techniques for determining the development of new treatments and measures for the prevention of emergent viral diseases (Allgaier et al., 2023).

2.2.5. Electronic Health Records (EHRs)

Electronic Health Records (EHRs), originally termed clinical information systems, emerged in the market through Lockheed in the 1960s (Allgaier et al., 2023). Since then the systems have undergone many reconstructing to build a standard system for the industry. To enhance the quality of work, improve efficiency, and promote EHR adoption, in 2009, the US federal government regarding billions of dollars to support EHR adoption across all practices; in turn, more than 87% of office-based practices in the US had adopted EHR systems by 2015 (Char et al., 2018).

Large data from EHR systems with structured feature data are helpful for deep learning, such as medication refills and diagnoses indications for patient history (Char et al., 2018). This has led to the enhancement of data management, data retrieval, and overall quality of care and assisted physicians in diagnosis and management. It has also led to increased availability of health records for research, through features being made consistent across several datasets (Jadhav et al., 2019).

Another EHR benefit lies in its capacity to contain structured and unstructured data, comprising patient identifiers, medical history, examination results, and clinical notes (Jadhav et al., 2019). This data can then be processed by machine learning models to make decisions on patients' outcomes and patterns. For instance, it is possible to estimate the probability of post-surgery complications using Machine Learning algorithms, which will enable healthcare workers to prevent such occurrences and enhance patient experiences (Mustafa and Rahimi Azghadi, 2021).

Another application of ML is its use in estimating readmission rates for patients using data collected from EHRs (Ahmad et al., 2020). Using patient data including history, treatment details, as well as socioeconomic factors predictive ML models can assess potential readmission risk and suggest how this risk might be mitigated. This not only has the advantage of improving patient health outcomes but also in lowering costs by decreasing readmissions. Another use of machine learning in EHRs is the ability to forecast disease prognosis. For instance, ML models can also be used to discover patterns in EHRs in the development of chronic diseases such as diabetes and heart disease (Gupta and Sedamkar, 2020).

Machine learning is also valuable in developing patterns that could be the same as those of patients with certain diseases or complicated diseases that can be hard to diagnose with traditional methods (An et al., 2023). Due to its ability to evaluate big data, the ML models can capture patterns and complexities not easily noticed by clinicians. It is especially useful in the identification of rare diseases which if detected in their early stages can be controlled. However, on a positive note, machine learning is improving the flow of operations in the healthcare industry through the elimination of mundane jobs like data input and invoicing. This ability to reduce the administrative burden makes the work of healthcare providers more focused allowing for the delivery of high-quality service in the hospitals and healthcare centers (Arvindhan et al., 2021).

2.3. Machine Learning in Pneumonia Detection

Pneumonia has been found to cause a raised mortality rate in children as young as five (Toğaçar et al., 2020). The best way to diagnose a patient for pneumonia is by the X-ray picture, which is more affordable and comprehensive as compared to other conventional diagnostic techniques. Projection within X-ray images for pneumonia takes a lot of time, and several radiologists might disagree on the disease prognosis. Therefore, this issue leads to the designing of pneumonia detection techniques that should be safe to employ in the health care department for real-time and accurate diagnosis of pneumonia (Chandra and Verma, 2020).

In disease diagnostic systems, ML, DL, and statistical methods are very efficient tools to be used (Chandra and Verma, 2020). They could be employed to tackle increasingly complex vision problems in the healthcare imaging segment, including but

not limited to lung disease categorization, lung segmentation, etc. The recent progression in the field of DL has been able to provide and possibly enhance human capability in many endeavors. DL can also be used to find out the impacts of treatments such as chevalier studies and cancer treatment. Labeled data and DL-based algorithms are related to challenging outcomes of thoracic disease classification with an X-ray image modality. In the past, deep neural network (DNN) models have been designed, implemented, and evaluated through a conventional method, which involves the use of a time-consuming trial-and-error technique by professionals in the field (Kareem et al., 2022).

2.3.1. The Role of Machine Learning in Medical Imaging for Pneumonia Detection

Chest X-rays have been a significant part of pneumonia diagnosis for decades as they help to evaluate the condition of the lung (Kareem et al., 2022). Chest X-rays and computer tomography (CT) are the most frequently employed imaging techniques for this purpose. However, these images are challenging to interpret, and even experienced radiologists sometimes find it hard to differentiate pneumonia from other related conditions, including bronchitis, pulmonary edema, or COPD (chronic obstructive pulmonary disease). This is even more difficult in patients who present mild radiological features of pneumonia or those who have underlying lung diseases that may alter the radiographic appearance of pneumonia (Al Mamlook et al., 2020).

ML, specifically the use of DL such as CNNs (Convolutional Neural Networks) has enhanced more of the field of medical imaging through the automatic analysis of these images (Al Mamlook et al., 2020). CNNs, developed to extract and analyze the features of images, can be trained on millions of labeled medical images and therefore detect aspects associated with pneumonia that might be inconceivable to the human eye. The efficiency of CNNs in handling a large number of images and in applying standard parameters to each case adds to its most important application where fatigue and cognitive errors can be expected in radiologists, especially in busy clinical facilities (Tsai and Tao, 2019).

For instance, in a seminal paper, the authors designed a CNN-based model backed by more than 100,000 chest X-ray images, which included pneumonia cases, normal, and other diseases like affected lungs (Tsai and Tao, 2019). Testing of the model yielded a

diagnostic accuracy of over 90% thus making it better than the conventional techniques. This is important, especially in clinical practice where accurate diagnosis is likely to determine the early intervention that shall have a major impact on the patient's condition. However, this machine learning model was not only identifying the presence of pneumonia but also separating it from other respiratory diseases, thus minimizing the likelihood of false positives and false negatives. This is quite crucial to eliminate wrong diagnoses and the subsequent unnecessary treatments that a patient may undergo (Yaseliani et al., 2022).

Besides improving diagnostic performance, machine learning models can also help in determining the severity of pneumonia based on the degree of lung involvement (Yaseliani et al., 2022). This entails assessing lobes that are affected and evaluating other factors including density in a lung, or consolidation which is characteristic of pneumonia in imaging setups. Using machine learning models, clinicians can determine the level of care patients need based on an accurate appreciation of disease severity. For example, if a patient requires a lot of lung regions to be flooded, that patient must be admitted or even put in the Intensive Care Unit, whereas a patient with mild disease can be treated as an outpatient. The convenience of the rapid and accurate identification of the degree of illness is most important in a condition where time is critical such as in severe pneumonia where the disease may progress to a critical state (Pankratz et al., 2017).

Additionally, in the case of machine learning, one can feed the model more data through training and in this way, the model evolves along with new data (Pankratz et al., 2017). This flexibility is quite useful in the ever-changing medical field through diseases and even the symptoms they manifest. For instance, during the COVID-19 outbreak, the models were quickly repurposed for the differentiation of COVID-19 pneumonia from other types of pneumonia. These models were, therefore, capable of adapting as they continue to learn from new data thereby remaining useful even as the pandemic advanced, and new mutations/variations of the virus were identified (Varshni et al., 2019).

2.3.2. Enhancing Diagnostic Accuracy with Machine Learning

The primary advantage of applying machine learning to pneumonia detection is a direct increase in diagnostic performance (Varshni et al., 2019). Timely and accurate diagnosis is highly valuable in medicine, as inaccurate diagnosis may lead to inadequate

treatment, slow healing, or even fatal consequences. Although traditional methods used in the diagnosis of pneumonia are efficient, they present a few challenges. These methods greatly depend on the proficiency and expertise of the radiologists, who are expected to assess intricate images and diagnostic information. Human interpretation is subjective and may differ from one clinician to another hence varying in diagnosis. However, human interpretation can be influenced by several factors like fatigue, biased thinking, and high workloads, which reduce accuracy even more (Das et al., 2022).

These challenges are eliminated for more objective, consistent, and scalable machine learning models in diagnosing pneumonia (Das et al., 2022). These models are useful in finding minute patterns that may not be visible in the images, especially in medical images. For instance, in patients presenting with early or mild pneumonia, there may be relatively insignificant changes on computed tomography images of the lung tissue and thus may not be observed even by prominent radiologists. Even the slight changes in tissues that cannot be easily seen and diagnosed by the human eye, machine learning algorithms especially those trained on large and diverse data sets can identify them and this can lead to early diagnosis (Račić et al., 2021).

Apart from enhancing diagnostic performance, machine learning models can process multiple forms of data at the same time, something that can be scarcely done by human clinicians (Račić et al., 2021). The diagnosis of pneumonia often involves the use of clinical data, imaging tests, and/or laboratory findings. For instance, a man with coughing, fever, and difficulty breathing will be advised to take a chest X-ray and a blood test to check whether the white blood cell count is high, which shows infection is present (Yee and Raymond, 2020).

In the past, the clinician would have to integrate this information qualitatively and quantitatively, a process that is both time-consuming and error-prone (Yee and Raymond, 2020). The machine learning models however can quickly assimilate and combine all these different types of data to give us a nearer real picture of the patient's condition. This not only brings a positive contribution to the diagnostic evaluation but also contributes to the existing diagnostic arsenal, optimizing the diagnostic and therapeutic algorithm, and bringing it to a higher level (Tilve et al., 2020).

An analysis of how machine learning has been applied in the identification of pneumonia demonstrates that these methods can enhance the diagnosis of respiratory

diseases (Tilve et al., 2020). In a work, scientists designed an image-overlaid model that incorporated chest X-rays and patient information, like symptoms and lab results, to estimate the risk of pneumonia. The model also performed far superior to traditional diagnostic methods giving a better and more precise diagnosis. Furthermore, the model proved to help identify the factors that contributed to the diagnosis which also involved concrete characteristics in imaging data or certain clinical signs. This is especially handy in multiple-factor cases where the nature of the disease or the injury needs to be determined after taking into account several factors (Barakat et al., 2023).

In addition, machine learning models are capable of learning new things over and over again, thus embracing what is known as 'continuous learning', and especially in the medical field where the occurrence of new diseases, new treatments, and diagnosis tools are always being invented (Barakat et al., 2023). They are more effective in diagnosing since they update their algorithms from time to time when new information is obtained. For instance, if there are novel subtypes of respiratory viruses as they occur, the models can be retrained based on data from the latest cases so that they remain relevant due to ever-evolving respiratory viruses (Muhammad et al., 2021).

The potential to improve the diagnostic performance of machine learning models is not only hypothetical but has been evidenced in numerous cases (Muhammad et al., 2021). For example, the diagnosis of pneumonia due to SARS-CoV-2 has been quickly diagnosed using machine learning models during the COVID-19 outbreak. These models were capable of distinguishing COVID-19 pneumonia from other lung pathologies despite having the same appearance on a chest radiograph or CT scan. Such precision helped deal with the pandemic because identifying the development of new symptoms and equipping the healthcare system to adapt to the same was never an easy task (Yahyaoui and Yumuşak, 2021).

2.3.3. Machine Learning in Predicting Outcomes and Guiding Treatment

In addition to diagnosis, machine learning models are also being applied for the prognosis of patient status and monitoring treatment decisions in cases of pneumonia (Yahyaoui and Yumuşak, 2021). It is essential to be able to predict pneumonia progression as well as to recognize patients who are most likely to develop complications,

to be able to provide the most effective treatment and appropriately distribute all the available healthcare resources (Swetha et al., 2021).

Machine learning models can use data obtained from imaging studies, EHRs (electronic health records), and/or patient demographics to assess outcomes like the need for mechanical ventilation, ICU (Intensive Care Unit) admission, or death (Swetha et al., 2021). Such predictions can assist healthcare providers in sorting patients according to their risk level so that the high-risk patients are well cared for (Liu et al., 2020).

For example, a machine learning model for predicting outcomes in patients with pneumonia used images and details of 30,000 over cases, radiographic, clinical factors, and non-urgent parameters (Liu et al., 2020). It also affords the chances of early interventions in those identified to be at higher risk of developing severe complications like respiratory failure or death. It also poses considerable consequences on prognosticating these outcomes with a high level of performance when it comes to resource allocation in care delivery structures, more so during a pandemic or outbreaks when resources are likely to be rationed (Barhoom and Abu-Naser, 2022).

Machine learning also provides an option for developing an individualized approach to the treatment of patients with pneumonia (Barhoom and Abu-Naser, 2022). Using individual patient data, such as genetics, other diseases, and previous responses to treatments, machine learning algorithms can point to the quintessential treatment for every patient. Such an approach is likely to benefit treatment by increasing its chances of success as well as minimizing the likelihood of side effects or other complications (Stokes et al., 2021).

Furthermore, the accuracy of machine learning models can be improved with increased robustness over data and time (Stokes et al., 2021). This ability to “learn” from new cases makes the diagnostic model especially useful for infected diseases where the diagnostic models are still being developed for new variants or new strains. As the disease dynamics change and new data becomes available, ML algorithms can therefore continue to provide accurate results (Gabruseva et al., 2020).

2.3.4. The Integration of Machine Learning with AI and Computer Vision

Machine learning has been further enhanced with artificial intelligence (AI) and computer vision in improving the detection of pneumonia (Gabruseva et al., 2020).

Computer vision, which is part of AI that underlines machines in interpreting visual info in the world, has been helpful in advanced imaging analysis tools' creation. Used with machine learning, computer vision can improve the sensitivity and specificity of pneumonia diagnosis, especially, when dealing with medical images like chest X-rays and CT scans (Cheekuri et al., 2024).

Pneumonia is one of the many areas that concern computer vision and its practical implementation, for instance, by creating image segmentation technologies (Cheekuri et al., 2024). Image segmentation is a division of one image into parts or segments and is useful when it is necessary to separate some areas of the image in particular. When aiming at pneumonia detection, segmentation tools help allow for distinguishing as well as outlining the lung consolidations that are signs of pneumonia in radiology. Pneumonia detection is accomplished by using these tools as they operate on machine learning to categorize the image and subsequently mark the lung tissue that may be potential pneumonia regions (Rahman et al., 2020).

There are several advantages of automated segmentation compared to the manual method employed by radiologists (Rahman et al., 2020). Primarily, it decreases the time needed for image analysis, and thus, the needed time for providing a necessary diagnosis and treatment. This is especially so where the patient is suffering from severe pneumonia for which early treatment has a significant impact on the patient's condition. Second, automated segmentation is completely different from human interpretation because it offers a more reliable and reproducible result, which minimizes inter-observer variability. This regularity assists in standardizing what each case is being judged against, therefore, creating higher levels of reliability and reproducible outcomes (Qu et al., 2022).

Apart from the segmentation, computer vision has also been employed to build machine learning models that would be used to differentiate cases of pneumonia that could be of different types based on the images taken (Qu et al., 2022). Community-acquired pneumonia can result from bacteria, viruses, and fungi infections and some types of pneumonia may have different radiographic imaging appearances. Pneumonia can be classified into different categories hence it is vital to determine the right type upon admission so that the right treatment plan can be administered. For instance, bacterial pneumonia may be managed by antibiotics while viral pneumonia may need antiviral medications or be supported with other means (Pap and Hrnčić, 2019).

Computer models or algorithms that use artificial intelligence may help in the diagnosis of bacterial, viral, and other forms of pneumonia by analyzing imaging data or scan results (Pap and Hrnčić, 2019). For instance, a model might interpret a chest X-ray image and infer typical features of bacterial pneumonia, including lobar consolidation or pleural effusion. It can then predict whether the pneumonia is bacterial and advise on the right treatment to be taken. The reason for such specificity is crucial in the proper identification of the disease and therefore the right treatment to avoid further complications (Absar et al., 2022).

In addition, the application of machine learning and computer vision has given rise to risk index models that can predict the likelihood of serious conditions associated with pneumonia (Absar et al., 2022). The aftermath of pneumonia includes pleural friction, lung abscess, and septic-driven acute respiratory distress syndrome (ARDS). Those are some of the complications of the illness and can be detected early so that the appropriate measures are taken to avoid the worsening of the patient's condition (Erdem and Aydin, 2021).

Imaging data integrated into the patient's electronic health record can feed into predictive models, such as computer vision, for the identification of possible complications like effusion in the pleural space, or lung necrosis (Erdem and Aydin, 2021). Such models can then generate a risk profile for clinicians, thereby showing which of the patients is at a higher risk of complications (Jain et al., 2022).

AI and computer vision when integrated with machine learning have also led to the creation of telemedicine applications for pneumonia diagnosis (Jain et al., 2022). Telemedicine is a concept that has received much attention recently, especially due to the occurrence of the COVID-19 pandemic. In areas where access to trained radiologists for interpreting chest X-rays is lacking, telemedicine platforms that employ machine learning algorithms can accurately interpret images (Kundu et al., 2021).

This capability is especially important in a rural or developing country setting to provide specialized medical services that may not be readily available in the area (Kundu et al., 2021). For instance, a telemedicine system may enable a healthcare provider from a distant rural area to take a chest X-ray image with a portable X-ray device or a smartphone. The picture can be transferred to a cloud-based server for analysis using a machine-learning algorithm for pneumonia indicators. Within minutes, the doctor or the

healthcare worker gets a diagnosis and can be able to decide on the next course of action for the patient. Implementing this approach enhances the general availability of diagnostic facilities while also guaranteeing that patients are promptly and correctly diagnosed irrespective of their geographical location (Goyal and Singh, 2023).

2.3.5. Machine Learning and Public Health: Addressing Pneumonia in Low-Resource Settings

Pneumonia continues to be a major and life-threatening disease, which is more or less prevalent in developing countries due to a lack of healthcare facilities and diagnostic equipment (Goyal and Singh, 2023). Thus, in such contexts, machine learning can be considered an effective solution in strengthening the diagnosis of pneumonia and decreasing mortality rates (GM et al., 2021).

Another way in which machine learning can be advantageous in low-resource environments is that it can work with little support structures in place (GM et al., 2021). In remote areas, machine learning models can be made available using portable devices such as mobile phones or tablets. Such models can even detect the conditions from portable X-ray or smartphone captures, making fast and correct pneumonia diagnoses in the field (Nimbolkar et al., 2022).

Another aspect of machine learning is that the use of such an approach will enable the diagnosis of pneumonia to be cheaper (Nimbolkar et al., 2022). The conventional procedures of diagnosis include laboratory diagnostic tests and diagnostic imaging and these are costly in terms of money, equipment, and personnel. After development, the utilization of machine learning algorithms for diagnosing patients would not require the additional utilization of the training data and can therefore be considered to be cost-effective for healthcare facilities that are constrained by financial resources (Naz et al., 2020).

Besides, it is possible to train machine learning models on population samples and enhance their performance on various groups of patients with pneumonia (Naz et al., 2020). It is accentuated particularly in low-income countries where the manifestation of pneumonia might not be akin to those observed in the developed world. Thus, there is a possibility of training models with locally appropriate data that allow healthcare providers

to think through the machine learning algorithms that would benefit their populations (Erdem and Aydin, 2021).

Machine learning also has the capacity for enhancing surveillance as well as response to disease outbreaks in low-income communities (Kundu et al., 2021). Machine learning is capable of capturing images, clinical records, and environmental data to discover pneumonia's outbreak and its potential trajectory. This information can support activities aimed at vaccinating the population or spreading knowledge about the disease, as well as the distribution of resources to address pneumonia outbreaks (Goyal and Singh, 2023).

2.4. Drawbacks and Ethical Problems in Using Machine Learning in The Healthcare Sector

2.4.1. Privacy and Data Protection

GDPR (General Data Protection Regulation) was first implemented by the EU and as other countries changed their privacy laws the amendments in those countries were made by GDPR (Khan et al., 2023). Based on these regulations, the controller or the union-based data processor also processes all personal data and the activities of foreign communities and companies to safeguard the information of natural persons with adequate protection. In the USA, the exists of organization known as the Genetic Information Non-discrimination Act (GINA) which discourages employers from making discriminative decisions based on the genetic health of anyone. Machine learning is used in the health care sector to process consumer health information, enhance diagnosis, and track the images of the medical devices that are used in patient treatment and care, and a supportive role in boosting the rate or pace of the related health research and development operations (Rasheed et al., 2022).

2.4.2. Bias and Discrimination

The performance of any machine learning model greatly depends on the input data used for training (Ballamudi, 2016). This implies that if the training data is biased or not generalized enough, the proposed models can contribute to the further enhancement of bias in the field of healthcare. For instance, machine learning models that a company uses

to make decisions may contain training data from a certain group of patients; thus, if a patient belongs to a different group, then they may be denied equal treatment and test results. These issues are especially poignant in a medicinal environment, as they could lead to misdiagnoses, improper treatment advice, and differentiated treatment. Bias can be mitigated in machine learning by ensuring the datasets used to train the model are carefully chosen and curated and by checking the model against data from multiple populations continually (Ngiam and Khor, 2019).

2.4.3. Informed Consent and Autonomy

Informed consent is one of the ways through which a patient and a health care provider communicate and encompass elements such as decision capacity and competency, documentation of informed consent, and ethical disclosure (Leimanis and Palkova, 2021). In this context, patients must have the right to be informed of their diagnosis and health condition, the treatment plan, the success of therapy, test results, costs, healthcare insurance share or other information related to medical activities, and any consent must be given for an exclusively specific purpose, freely and unambiguously. Fears for this problem emerged even with the advancement of machine learning in the applicative context of health (Prakash et al., 2022).

2.4.4. Impact on Healthcare Jobs

The tendency of automating jobs that were previously assigned to healthcare workers gives rise to the issue of job losses and downgrading of some abilities (Naik et al., 2022). However, it also suggests that machine learning can decrease the workload for healthcare providers and, at the same time, be a potential threat to some professionals as it may replace some of them and become an efficient tool for analyzing images or data, for example. This can result in the loss of jobs or a change in what is considered a desirable qualification in the field of healthcare. Their use also offers the prospect of depersonalization of care, where compassionate touch, language, and patient relations may be eradicated in a bid to make way for new efficient, and optimized technologies (Wan, 2022).

2.4.5. Medical Consultation, Empathy, and Sympathy

The idea of applying machine learning to all realms of health care appears challenging and unachievable (Naresh and Thamarai, 2023). Since human and medical robots may experience uniquely human feelings, they cannot develop together shortly. Doctors and other caregivers should consult or be consulted by another doctor or another caregiver, which is impossible with smart or robotic machines. On the other hand, patients will not accept 'machine-human' medical relations than the 'human-human' (Siddiq, 2021).

Doctors and nurses are supposed to treat patients with the touch of empathy and compassion needed to treat the patients, which is likely to mar the healing process of the patients (Kelly et al., 2019). This will not, however, be possible through robotic doctors and caretakers. Empathy, kindness, and compliance with appropriate behavior are qualities that patients will lack when interacting with robotic physicians and nurses since robots lack humane characteristics. These are considered some of the most tragic impacts of artificial intelligence and machine learning in medical science (Gabriel, 2023).

3. THE DATASET

Before diving into the model itself, it is Worth to pause on the dataset that have been used as data source.

To effectively train our model to predict whether a patient is suffering from pneumonia, we are utilizing a comprehensive dataset of chest X-ray images sourced from Kaggle (*Chest X-Ray Images (Pneumonia)*, 2018). It consists of anonymized patient chest X-ray pictures that have been classified as either "Normal" or "Pneumonia" according to the patients' health conditions. There are 2,615 images in the 'Normal' category and 4,811 images in the 'Pneumonia' category for the training set. There are 234 pictures of healthy patients and 390 pictures of pneumonia sufferers in the test set.

Despite the large number of elements in the dataset, data augmentation is required to improve the model performance. By using data augmentation approaches, the resilience and generalization abilities of the model will be enhanced by diversifying the training set without having to gather additional data. The details of data augmentation will be covered in detail in the upcoming sections.

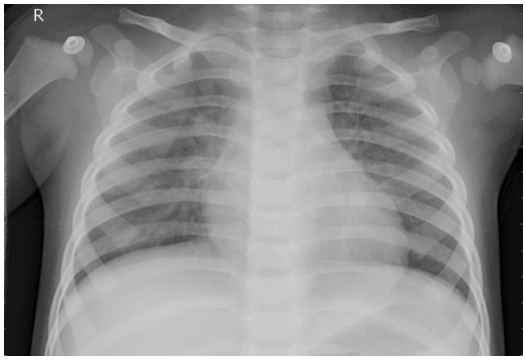


Image 3.1 Pneumonia

Image 3.2 Normal

Our pneumonia detection model's accuracy and reliability can be greatly increased by making sure the dataset is well-prepared and enhanced.

For both Training and test set images, the pixel dimensions are not consistent because the pictures are real, legitimate X-rays. The following tables provide a summary of the dimensions' average and standard deviation (SD) for the training and test images:

Training Set		
	Pneumonia	Normal
Average Width:	1202.03	1666.17
Average Height	827.58	1378.28
Width Standard Deviation	284.23	297.19
Height Standard Deviation	272.43	335.04

Test Set		
	Pneumonia	Normal
Average Width:	1140.82	1800.30
Average Height	765.29	1369.09
Width Standard Deviation	208.59	365.15
Height Standard Deviation	192.85	428.26

(you can find the code used in appendix I)

As detailed in the previous tables, the dimensions of the images exhibit considerable variability. However, this variability is not a substantial issue because the models to be utilized, specifically ResNet and VGG16, necessitate fixed dimensions for input images. Consequently, the images will be resized prior to training each model. It is fundamental to assess any possible issue before proceeding with the resizing process, as altering the dimensions of images can sometimes lead to distortions, thereby impairing the model's ability to generate accurate predictions. Fortunately, in this context, resizing the images does not induce to much distortion, thereby almost totally preserving the quality and informational content of the images as shown by the examples below.

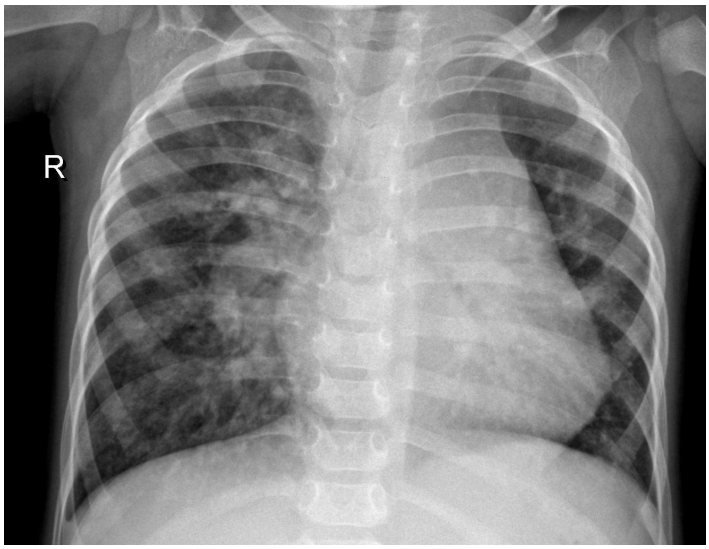


Image 3.3 Original Image

Image 3.4 Resized image

4. PREPROCESSING

4.1. Data Augmentation

Training a reliable machine learning model relies significantly on the size and quality of the training dataset. While there isn't a precise number that defines what is "enough," it is universally acknowledged that more data generally leads to better model performance. However, merely increasing the quantity of data is not sufficient; the diversity and quality of the data, so that the model is always fed with a different image each time, are equally important to ensure the model captures a wide range of variability and avoids overfitting.

Augmenting data in this case doesn’t mean creating images from scratch (even though it is also a suitable, but more costly, approach) but applying some transformations to images that are already part of the initial dataset. In most of the cases these transformations are rotation, flipping, scaling, change the brightness level and so on.

In order to perform data augmentation it has been used a function that is part of the keras library: ImageDataGenerator. As described by Chollet, B. F. (n.d.) using Keras ImageDataGenerator the following transformations can be applied to the images:

- rotation_range: it is a value in degrees (0-180), a range within which to randomly rotate pictures
- width_shift and height_shift: they are ranges (as a fraction of total width or height) within which to randomly translate pictures vertically or horizontally
- rescale: it is a value by which we will multiply the data before any other processing. Our original images consist in RGB coefficients in the 0-255, but such range of values would be too high for our models to process (given a typical learning rate), so we target values between 0 and 1 instead by scaling with a 1/255. factor.
- shear_range: it is for randomly applying shearing transformations
- zoom_range: it is for randomly zooming inside pictures

- horizontal_flip: it is for randomly flipping half of the images horizontally, relevant when there are no assumptions of horizontal assymetry (e.g. real-world pictures).
- fill_mode is the strategy used for filling in newly created pixels, which can appear after a rotation or a width/height shift.

And in particular the model used for answering the thesis question has used this set of values for the above mentioned parameters: rescale=1./255,

rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, vertical_flip=True,

brightness_range=[0.8, 1.2], fill_mode='nearest'.

As can be noticed the transformations are applied but they are not heavy so that input images are not too skewed.

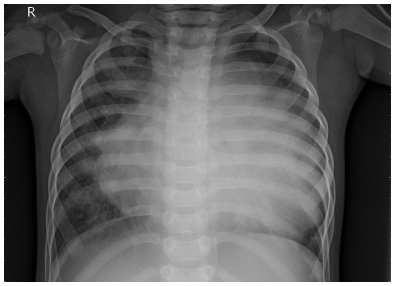


Image 4.1.1 Original Image

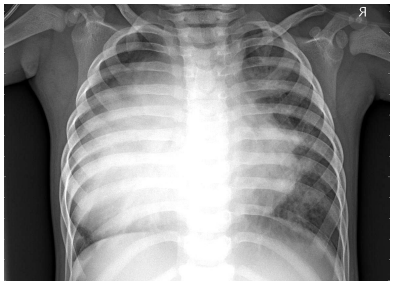


Image 4.1.2 Different brightness

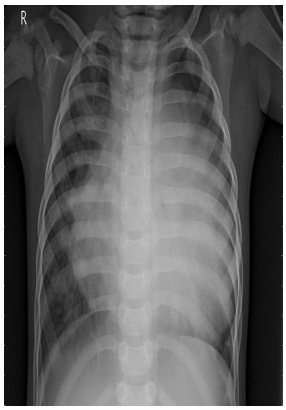


Image 4.1.3 Height shift range

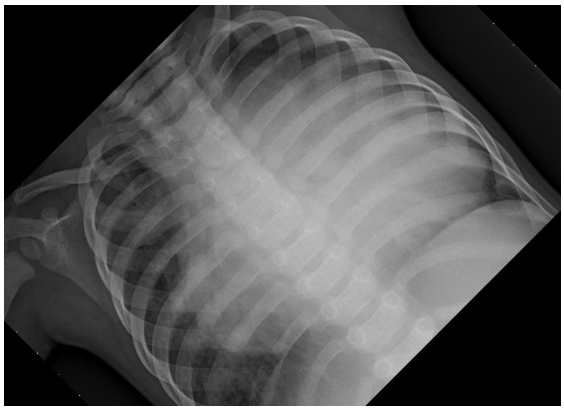


Image 4.1.4 Rotation

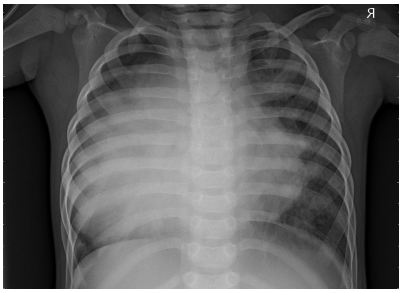


Image 4.1.5 Flipped

However, before applying data augmentation, it is worth to pause on a possible problem that might arise from this practice. In classification problems, either binary or multiclass, having an imbalanced dataset is very risky and can result in some issues; as explained by Jason Brownlee in his article about imbalanced classification (2022), "Imbalanced classification is the problem of classification when there is an unequal distribution of classes in the training dataset." In the analyzed scenario the risk is to have more images of Pneumonia cases then Normal cases or vice versa.

Several risks can arise from this situation like:

- Poor generalization: the model could not generalize well on unseen data especially if these data belong to the least represented class;
- Bias towards majority class: the model could perform very well in the predicting the majority class while being very poor in predicting the minority class;
- Skewed predictions: the model would more easily predict a test sample to belong to the majority class rather than to the minority class.

In order to avoid one of the just mentioned problems, a multiplier has been set to each category. Since the images belonging to the Pneumonia class were much more than the images belonging to the Normal class, for each pneumonia image 2 transformed images were created while for each normal image, 4 transformed images were created.

These 2 ratios were selected since only integer numbers were accepted by the functions, but since this approach results in having more normal images than pneumonia ones, a random sample of normal images has been selected and erased to balance the two classes.

The code used is available at Appendix II

4.2. Validation set

When training a ML model, principally CNNs, splitting datasets into training and validation sets is an essential practice, typically allocating around 30% of data for validation, allows for the assessment of model performance on unseen data, detection of overfitting, and fine-tuning of hyperparameters. Hyperparameters, such as learning rate and batch size, are optimized using the validation set (Academy 2023).

Because of this, 30% of the images belonging to both the Pneumonia and Normal category have been copied and allocated to the validation set.

Take a look to appendix III to see the code

4.3. Modifying test set

Before proceeding with more technical steps, one last task was to modify the naming convention of the test images. Originally, these images included labels indicating the cause of Pneumonia (virus, bacteria) or the term 'normal.' Such labels could inadvertently introduce bias, allowing the model to use this information to aid itself in making predictions, thereby skewing the results. To ensure the predictions were based solely on the image data and to maintain the integrity of the evaluation process, it was crucial to remove any identifying information from the test image filenames. This modification helped preventing the model from leveraging non-visual clues, resulting in a more accurate and unbiased assessment of its performance.

5. DATA LOADERS

Because of what have been said previously about the size of the input dataset for a machine learning model, making a PC handle that big amount of images can be risky: the risk is that the dimension of the whole set of images exceeds the available RAM of a single computer and this could result in two possible scenarios, the constant reading/writing operations between RAM and disk could severely slow down the whole process or ,even worst, the computer is not able to keep that amount of images in its RAM all at the same time and the process will fail, making it impossible to go forward.

Luckly, during past years, several tools have been implemented to overcome this issue; in this case the choice fell on PyTorch Data Loaders.

PyThorch, together with Keras, is one the most used and known python library for deep learning, created in 2016 by the Facebook's artificial intelligence research team and it is an open-source machine learning platform. It is built on the Lua-written Torch library, which is a framework for scientific computing (Educative, n.d.).

In PyTorch, DataLoader is a built-in class that provides an efficient and flexible way to load data into a model for training or inference and it is particularly convenient because not only it helps in dealing with data that cannot fit into local RAM but it also allows to perform some data preprocessing and shuffling inside itself (Educative, n.d.).

The PyTorch DataLoader class is a utility class, that exactly do what is needed in our scenario, it loads training data in batches of fixed size and feed them to the model.

DataLoader is a class part of the PyTorch data loading library, which includes other classes such as Dataset, Sampler, and BatchSampler; these classes work together to create efficient and flexible data loading pipelines for deep learning models (PyTorch DataLoader: Features, Benefits, and How to Use It | Saturn Cloud Blog, 2023).

As it could have been possible to grasp, there are some advantages in using PyTorch DataLoader, let’s summarize them:

- Efficient data loading: The DataLoader class provides efficient data loading by allowing the user to load data in parallel using multiple CPU cores. This can significantly reduce the data loading time, especially for large datasets.

- Flexibility: The DataLoader class is highly flexible and can handle a wide variety of data formats and sources. It supports loading data from files, databases, and other external sources, as well as pre-processing data using custom functions.
- Shuffling: The DataLoader class provides shuffling capabilities, allowing the user to shuffle the data for each epoch. This can help prevent the model from overfitting to the training data and improve its generalization.

(PyTorch DataLoader: Features, Benefits, and How to Use It | Saturn Cloud Blog, 2023).

PyTorch DataLoaders are architecturally complex, it can be useful to go through some of their components.

A “dataset” in PyTorch is an abstract class representing a collection of data. It is responsible for loading and preprocessing data from a source and returning it as a PyTorch tensor. The Dataset class provides two main methods: __len__, which returns the length of the dataset, and __getitem__, which returns a single data point from the dataset at a given index. The __getitem__ method is crucial as it handles the actual data loading and preprocessing. This method is used by DataLoaders to load and preprocess the data efficiently.

The “DataLoader “is a utility class in PyTorch that facilitates iterating over a Dataset object in batches. It is designed to handle large datasets efficiently and can be configured to load data in parallel, preprocess data on the fly, and shuffle data for each epoch. By taking a Dataset object, the DataLoader offers various configuration options, including batch size, shuffling, and the number of worker processes for parallel data loading. It is responsible for batching the data and returning it in a format that can be readily consumed by the model.

A “Sampler” in PyTorch determines which samples should be included in each batch by using an indexing strategy. The most common types of samplers are SequentialSampler, which samples data in sequential order, and RandomSampler, which samples data randomly.

“Transformations” are an integral part of the PyTorch DataLoader architecture, applied to the data during loading to preprocess it for use by the model. The built-in transformations that PyTorch offers are several like resizing, cropping, normalization, and data augmentation. These transformations can be applied either during loading by the Dataset class for fine-grained control or during batching by the DataLoader class to improve performance by reducing the preprocessing load.

Going on, the “batch” is the basic unit of data, typically a tensor of shape (batch_size, input_shape), where batch_size is the number of data points in the batch (number of images that are loaded in the RAM at each step) and input_shape is the shape of the input data. The DataLoader is responsible for batching the data from the Dataset object and returning it in a format suitable for model consumption. The batch size can be specified creating a DataLoader object.

To conclude, “Shuffling” is the process of randomly reordering the data in a dataset to prevent the model from overfitting to the data order. The PyTorch DataLoader class allows for data shuffling for each epoch by setting the shuffle parameter to True. When enabled, the DataLoader randomly shuffles the indices of data points, returning the data in a random order for each epoch.

In this case study, for each of the implemented model, the DataLoader architecture was quite similar the only difference was the input size of the images since each model was requiring an exact value; this said the other parameter were set to: shuffle = True, batch_size = 64 and inside the transformation parameter the images were set to the correct input shape, kept with 3 color channels and their pixels values normalized.

The full code used can be assessed in appendix IV.

6. MODELS

The main focus of this work is the one understanding how much reliable ML models are to predict whether a patient is suffering from Pneumonia or not and which model performs the best. Three different models will be tested and discussed: ResNet,VGG-16 and Optimized CNN.
There will be a dedicated chapter for all of them where we will be going through their structure and they training process, but it Is worth to spend some words on CNN more in general.

6.1. CNN

Researchers in AI have been working to create systems that can comprehend visual input since the 1950s. This work led to the development of the field of computer vision. When University of Toronto researchers created AlexNet, an AI model that greatly surpassed earlier picture recognition systems, in 2012, a major breakthrough took place. Alex Krizhevsky's AlexNet won the 2012 ImageNet competition with an accuracy of 85%, easily outperforming the 74% of the runner-up. CNNs, a particular kind of neural network that imitates human vision, were the key to this achievement.
CNNs are becoming essential for computer vision tasks including segmentation, object identification, and picture classification (Mandal, 2024).
Differently from classical ANN, CNNs architecture uses a special technique called Convolution that doesn’t rely solely on matrix multiplications but combines two functions to show how one changes the shape of the other. The final target of CNNs is the one of reducing images into a form that is easier to process, without losing features that are critical for getting a good prediction (Mandal, 2024).
Since CNNs are a ML model, they are not able to read images as such, but they need to look at them as a set of numbers.

Image 6.1.1 (Chatterjee, 2021)

That considered in a matricial form it would result in something like that:

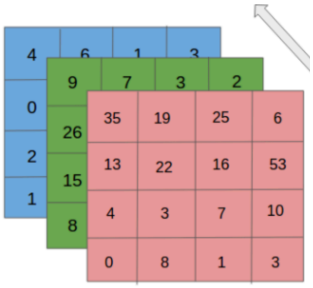


Image 6.1.2 (Mandal, 2024).

The different layers of which a CNN can be composed are Convolution layer, Pooling layer, flattening layer.

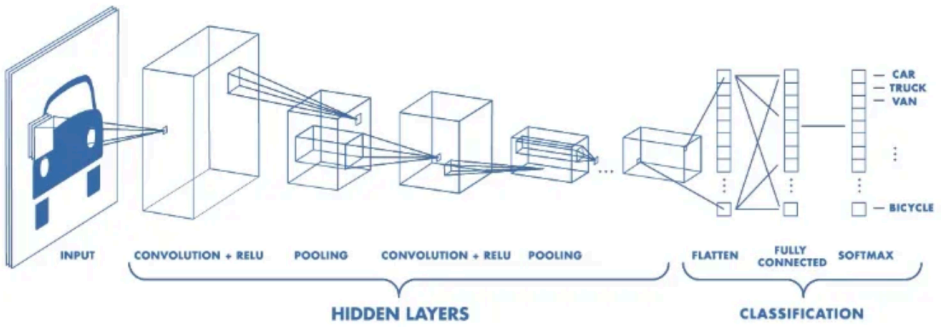


Image 6.1.3 (Mishra, 2021)

6.1.1. Convolution layer

The convolution layer is the core building block of the CNN. It carries the main portion of the network's computational load, applying systematically learned filters to input images in order to create feature maps that summarize the presence of those features in the input (*A Gentle Introduction to Pooling Layers for Convolutional Neural Networks*, 2019). This layer performs a dot product between two matrices, where one is called kernel (also called filter) and the second one is a matrix, of the same shape of the kernel, taken from the raw image. If the image is composed of three (RGB) channels, the kernel height and width will be spatially small, but the depth extends up to all three channels (Mishra, 2021). The output of this dot product is called feature map. The role of the kernel is the one of trying to catch some features, lines, edges or anything that could be useful to perform the desired classification. At each epoch, after the back propagation, the values in the kernel matrix are changed accordingly to catch features are more relevant.

Two parameters related to the convolution layer, are stride and padding, from which depends on the size of the feature map. The stride represents how much the filter will move, horizontally and vertically, at each step, over the input image, the bigger the stride, the smaller the size of the feature map. The padding parameter gives the possibility to surround the input image, on the vertical and horizontal side, with a column or row of zeros; padding the image, as changing the value of the stride, can modify the size of the feature map.

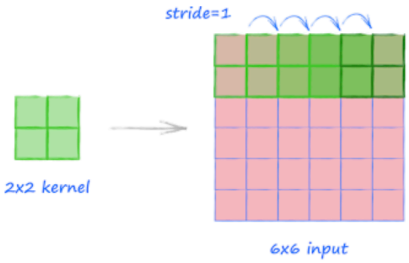


Image 6.1.1.1 (NeuralNet, n.d.)

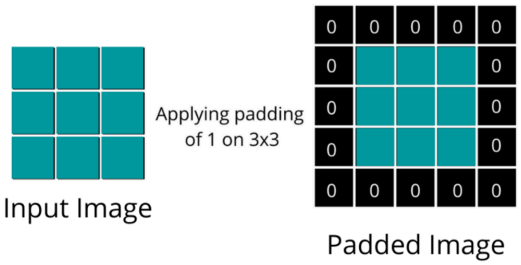


Image 6.1.1.2 (Chaudhary, 2023)

6.1.2. Pooling Layer

A limitation of the feature map output of convolutional layers is that they record the precise position of features in the input. This means that small movements in the position of the feature in the input image will result in a different feature map. This can happen with re-cropping, rotation, shifting, and other minor changes to the input image. One approach to address this sensitivity is to down sample the feature maps. This has the effect of making the resulting down sampled feature maps more robust to changes in the position of the feature in the image, referred to by the technical phrase "local translation invariance." Down sampling can be achieved with convolutional layers by changing the stride of the convolution across the image. A more robust and common approach is to use a pooling layer. Pooling layers provide down sampling feature maps by summarizing the presence of features in patches of the feature map (*A Gentle Introduction to Pooling Layers for Convolutional Neural Networks*, 2019).

There are two types of pooling average pooling and max pooling. What it is done in Max Pooling is finding the maximum value of a pixel from a portion of the image covered by the kernel. Max Pooling also performs as a Noise Suppressant, it discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. Average Pooling simply performs dimensionality reduction as a noise-suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling

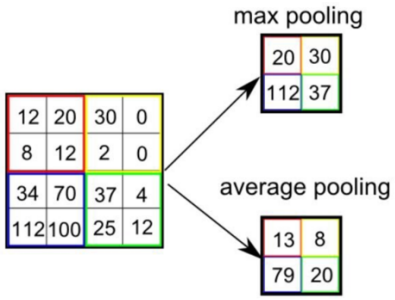


Image 6.1.2.1(Mandal, 2024b).

Beside choosing between max and average pooling another parameter to choose is the size of the dimension of the polling mask, that usually is set to 2x2 od 3x3. This consistently reduces the size of each feature map by half, halving each dimension and reducing the number of pixels or values in each feature map to one-quarter of its original size. For instance, a pooling layer applied to a feature map of 6×6 pixels (36 pixels) results in an output pooled feature map of 3×3 pixels (9 pixels). The pooled feature maps are a summarized version of the detected features, making the model invariant to minor positional changes in the input. This enhances the model's robustness to small positional shifts in the features detected by the convolutional layer.

6.1.3. Activation functions

Before deep-dive into the last two layers of the CNN it is important to pause on the activation functions that have been cited in the last paragraph. An activation function is a mathematical function applied to a neuron's output in a neural network. It decides whether a neuron should be activated by calculating the weighted sum of inputs plus a bias term and then applying the function to this sum. The primary purpose of an activation function is to introduce non-linearity into the network, enabling it to learn and represent more complex patterns than a linear model. Non-linearity allows the neural network to approximate complex functions and make decisions based on a combination of features.

If we only used linear transformations, the model would not be able to capture the underlying patterns in data that involve multiple interacting factors. Non-linear activation functions allow the network to learn these interactions and capture more intricate details of the data. In backpropagation, the backward pass uses the gradients of the activation functions to update the weights; without non-linear activation functions, the gradients would simply be linear transformations, preventing the network from learning anything other than linear relationships. (GeeksforGeeks, 2024).

Some of the most popular activation functions are:

- **Sigmoid (Logistic) Function:** $\sigma(x)= \frac{1}{1+e^{-x}}$

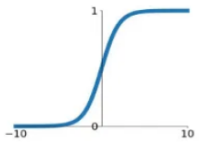


Image 6.1.3.1 (Jadon, 2022)

- Hyperbolic Tangent (Tanh) Function : $\tanh(x)=\frac{2}{1+e^{-2x}} - 1$

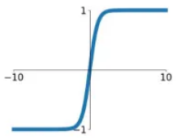


Image 6.1.3.2 (Jadon, 2022)

- Rectified Linear Unit (ReLU): max(0,x)

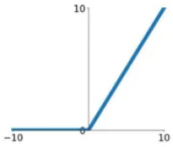


Image 6.1.3.3 (Jadon, 2022)

6.1.4. Flattening and Fully connected Layer

The next layer is the flattening layer, a preparation phase to the last layer, the fully connected one. Since the fully connected layer requires a precise shape of its input, what it is done in this layer is transforming the pooled feature map into a 1-dimensional array.

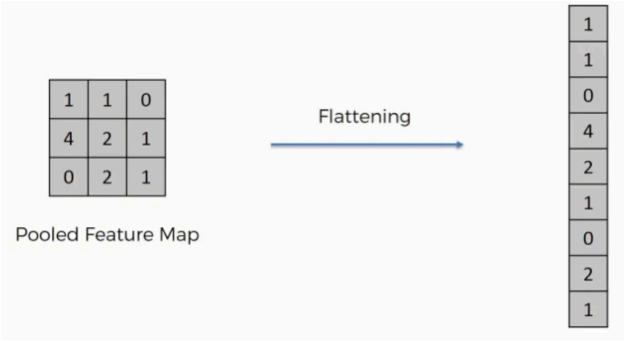


Image 6.1.4.1 (Ali, 2022)

The flattened array is the input for the very last layer of the CNN, the fully connected layer. This layer can be said to have the same shape of a classical ANN, where there are layers of neurons until reaching the last layer where the classification, based on the probability to belong either to a class or to another one, is performed. The peculiarity of this layer is that all the neurons of each layer are connected to all the neurons of preceding and succeeding layer.

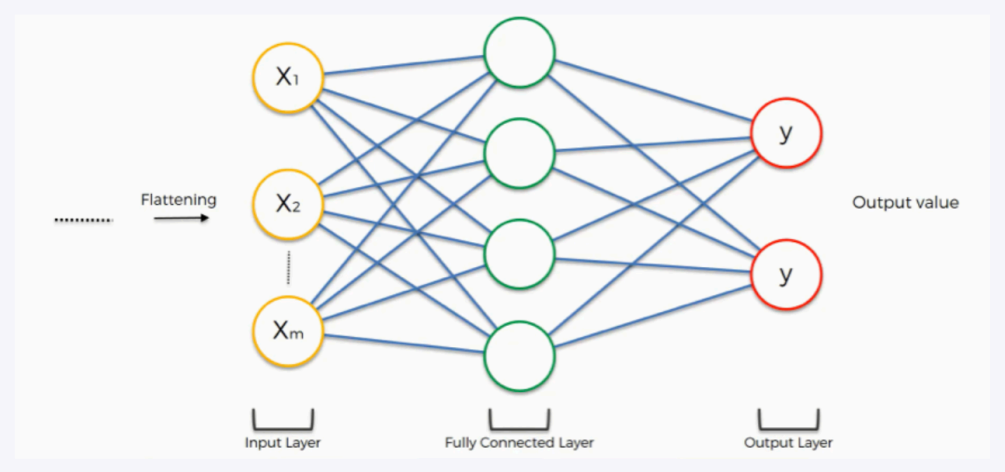


Image 6.1.4.1 (Convolutional Neural Networks (CNN - SuperDataScience.)

6.2. MODELS STRUCTURE

In this project, three different models, ResNet50, VGG16, and a tuned CNN were compared and analyzed. The following paragraphs will explain the functionality of each model and evaluate their performance to determine which one proved to be the best and the most effective in this particular case.

6.2.1. Transfer Learning

Before deep diving into the architecture of the model, it is worth to pause for a second how the concept behind ResNet and VGG16, two of the most used models for this type of problems: transfer learning.

Transfer learning is a machine learning and deep learning method in which a model that has been already trained on one task can be adopted for training another one. In transfer learning, the model is not initiated from scratch, instead, the knowledge learned by some other pre-existing situation is transferred to the target case to achieve better output.

Typically, this pre-trained model has been trained on a large-scale dataset like ImageNet to solve a general problem in computer vision; the learned features are then applied to the new task.

In transfer learning, one typically replaces the pre-trained model's final layer with a new layer made specifically for a given task. This new layer is then trained on the task-specific dataset, whereas the rest of the pre-trained model remains unmodified with fixed weights. This fine-tunes the model for the new task in a way that does not overfit and reduces training time (the benefits of this method will be explained more in depth in the following section). Transfer learning has become an important method for deep learning, which significantly increases accuracy and speed in many computer vision applications Vishal. (2024, January 29).

6.2.2. ResNet50

In the previous paragraph the structure of CNNs has been explained and it is precisely on CNN that the first model we are going through is based: ResNet50.
ResNet-50, developed by Microsoft Research in 2015, is a deep convolutional neural network renowned for its depth and the implementation of skip connections, or residual connections, which effectively tackle the vanishing gradient issue common in very deep networks. These skip connections enable the network to learn residual mappings by incorporating shortcut paths that bypass one or more layers, allowing it to concentrate on learning the difference (residual) between the input and output rather than the direct underlying mapping. This methodology facilitates efficient training of very deep networks by simplifying the learning process and mitigating the vanishing gradient problem (Rehman, 2024).

ResNet-50's architecture consists of 50 layers, making it a highly potent and efficient tool for image classification tasks. It is pretrained on the extensive ImageNet database, which includes over a million images across 1,000 categories, and contains more than 23 million trainable parameters. This substantial depth makes it particularly adapt to image recognition tasks. Leveraging a pretrained model like ResNet-50 offers significant advantages over building a model from scratch, as it reduces the need for extensive data collection and training. While other pretrained models such as AlexNet, GoogleNet, and VGG19 are available, ResNet-50 distinguishes itself through superior generalization performance and lower error rates in recognition tasks, making it an invaluable asset for image classification. (Danielsen, 2021)

In recent studies focused on pneumonia detection, deep learning models like ResNet-50 have been utilized to differentiate between normal and severe pneumonia cases. Pneumonia, a lung inflammation disease, often shares visual characteristics with other respiratory illnesses, posing challenges for traditional diagnostic methods. Other models have already been trained in order to predict whether someone was suffering from this type of inflammation or not; in one of these cases, a deep learning model was developed and evaluated on two datasets containing 5,856 and 112,120 chest X-ray images, respectively. The study assessed the performance of eight pretrained models, ResNet50, ResNet152V2, DenseNet121, DenseNet201, Xception, VGG16, EfficientNet,

and MobileNet, using metrics such as accuracy, precision, recall, and F1-score. Among these, the ResNet50 model achieved a very high accuracy, proving to be one of the most effective model for pneumonia detection in this research (Reshan MSA et al 2023).

Skip connections are features commonly found in many convolutional architectures. They provide alternative paths for the gradient during backpropagation, which can improve model convergence.

The update rule for gradient descent without momentum is given by the following equation:

$$w_i' = w_i + \Delta w_i$$

Where

$$\Delta w_i = -\lambda \frac{\theta L}{\theta \Delta w_i}$$

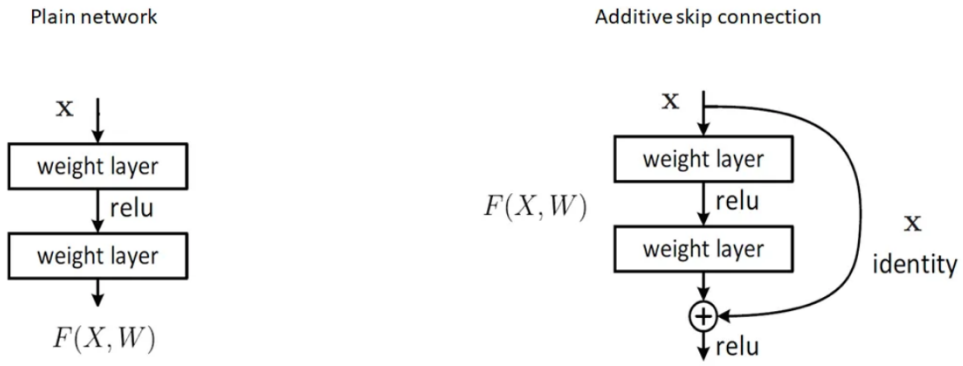
Updated parameters = Current parameters − λ · θL

Where θL is the gradient of the loss function L with respect to the parameters, and λ is the learning rate. Basically, the update rule slightly adjusts the parameters based on Δw_i, which is derived from the gradient of the loss function. The loss function quantitatively measures how far the model's predictions are from the actual target values. The goal is to minimize this loss function until it cannot decrease further or until a predefined stopping criterion is reached. To achieve this minimization, backpropagation iteratively updates the network's parameters.

A skip connection allows the output of one layer to bypass intermediate layers and feed directly into a deeper layer, rather than passing through each layer sequentially.

In deep networks, as gradients are backpropagated to earlier layers, they can become very small (vanish), which makes learning stagnant. Skip connections help by allowing the gradient to be directly backpropagated to earlier layers, maintaining a healthier gradient flow through the network.

That is exactly the most important advantage that skip connections provide in training. Deep neural network. (Kien, 2022)



On the left an example of the plan usual path of backpropagation, on the right an example of the path when using skip connections.
Image 6.2.2.1 (Cannata, 2021)

The core component of ResNet architectures is the residual block, comprising multiple convolutional layers that are interconnected through skip connections. The innovative aspect lies in the fact that skip connection combines the original input of a block with its output before the activation function is applied. This design allows for identity mapping when the input and output dimensions match, giving the network the flexibility to either learn the residuals or pass the input unchanged. This mechanism streamlines optimization and enhances the network's ability to effectively train deeper models.

In developing the ResNet50 model for our pneumonia detection purpose some points of attentions have been observed. Because the model to be trained efficiently has to receive images in a well-defined input shape, before feeding the deep neural network all the images imported by the DataLoader had been converted in a 250x250x3 shape and its pixels values normalized. The second point of attention that have been considered in the training process is about a structural feature of ResNet50: freezing and unfreezing final layers of the pretrained model.

It has been said previously that the model is pretrained on the ImageNet dataset, a dataset containing a large number of diverse images belonging to a considerable amount of categories. This comprehensive pretraining enables the model to learn a wide range of visual features applicable to various types of images. Using a pretrained model is more efficient, saving time and computational resources compared to building a new model from the ground up. However, the ImageNet dataset lacks of X-ray images, which have

distinct characteristics compared to natural images. To tailor the model to our specific task, segmenting patient suffering from pneumonia and patients that are healthy, a fine-tuning approach has been adopted. This method allows us to retain the general feature extraction capabilities of the pretrained ResNet50 while also modifying the model to better recognize the unique features of X-ray images. To pursue this scope, the model has been previously trained keeping its last layer frozen, maintaining the weights of the pretrained model leveraging its existing knowledge, and subsequently the last 10 layer were unfrozen and the model trained again to finetune the value of the weights. This step helps the model to better align with the specific features found in X-ray images and to learn more specialized patterns relevant to the new dataset, enhancing its performance on the task at hand. Keeping in mind what have been said in the previous point, another ruse to make the model adapt to X-Ray images is to build some more custom layers on top of the pretrained ResNet model. In this case two sets of layers have been constructed where each set was composed by a Convolution Layer (with kernel_size=3 and padding =1), a batch normalization layer (to make the training process faster and more stable), a layer with the ReLU activation function, a max pooling layer, a dropout layer (to reduce the risk of overfitting) and a fully connected layer at the end.

The code used to train the ResNet50 model can be analyzed in appendix V.

6.2.3. VGG16

VGG16 is a notable Convolutional Neural Network (CNN) architecture, which gained recognition after winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014. It is considered one of the most effective vision model architectures available today (Great Learning, 2022) (Thakur, 2024). The uniqueness of VGG16 lies in its simple and consistent design. Instead of using numerous hyperparameters, VGG16 employs a straightforward approach: it utilizes convolutional layers with small 3x3 filters and a stride of 1, consistently applying the same padding. Additionally, it incorporates max pooling layers with 2x2 filters and a stride of 2. This uniform pattern of convolution and max pooling layers throughout the architecture significantly enhances its

effectiveness (Great Learning, 2022b) (Vepuri, 2022) .The VGG16 architecture consists of 16 layers with trainable parameters, including 13 convolutional layers and 3 fully connected (dense) layers. While the network has a total of 21 layers, only 16 of them are weight layers. With approximately 138 million trainable parameters, VGG16 is a large network capable of learning complex patterns from data, making it highly effective for image recognition tasks. (Great Learning, 2022b) (Vepuri, 2022)

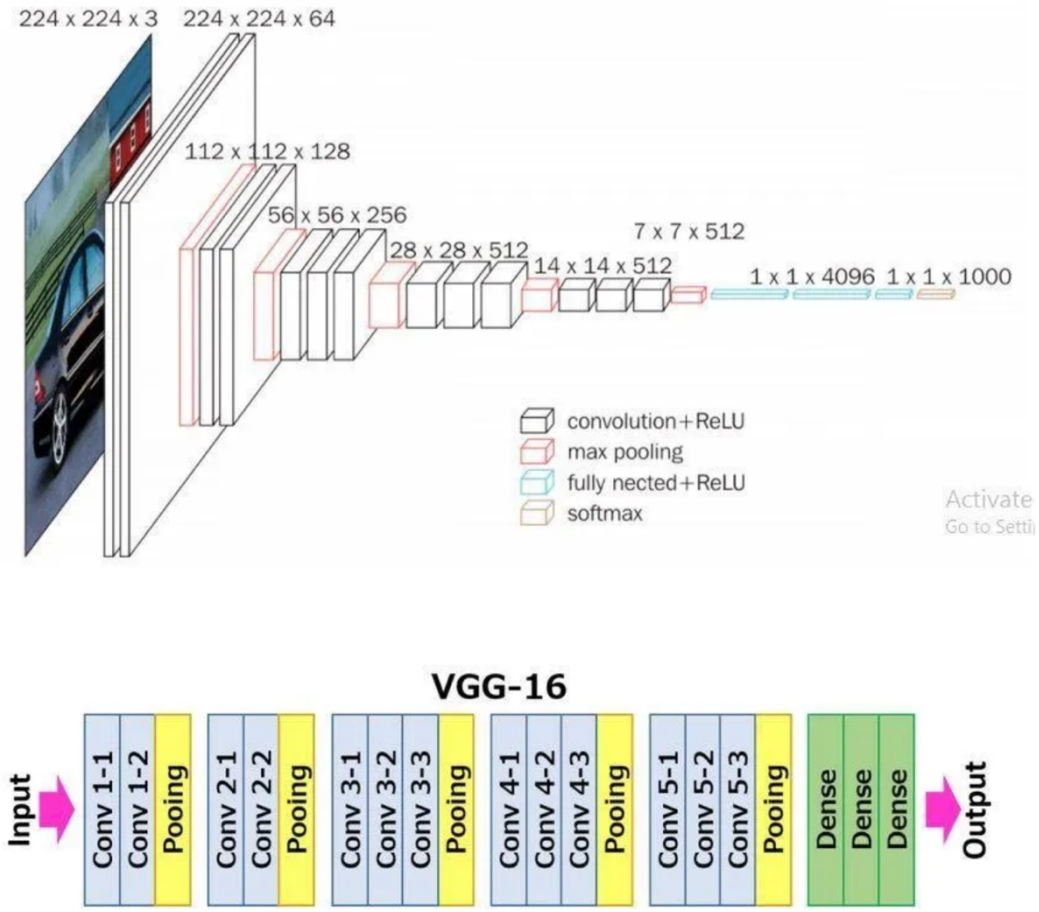


Image 6.2.3.1 VGG 16 architecture (Great Learning, 2022b)

The design of VGG16 emphasizes simplicity by consistently using small 3x3 convolution filters with a stride of 1. This strategy allows the network to capture fine details in images, enhancing its ability to learn and recognize complex features. All convolutional layers use the same padding to maintain the spatial dimensions of the input, ensuring that essential image information is preserved. Max pooling layers, which follow certain convolutional layers, reduce the spatial dimensions of feature maps while keeping

the most critical features, using a 2x2 filter with a stride of 2. The "16" in VGG16 refers to its 16 weight layers, which include both convolutional and fully connected layers. The depth of the network enables it to learn hierarchical representations of images, enhancing its performance on complex visual tasks. Compared to earlier models, the increased depth of VGG16 allows it to learn more intricate patterns and features from input data (Great Learning, 2022b) (Vepuri, 2022). As explained before for the ResNet50 model, also in building the VGG16 some diligences have been put on some points. The points of attention are the same as for ResNet50, indeed also in this case the last layer of the pretrained model have been kept frozen in first instance and unfrozen successively and images have been fed to the model in the proper shape 224x224x3.
If you are interested to the code used to train the VGG16 model go to appendix VI.

6.2.4. OPTIMIZED CNN

The third model built and compared with the other two is an optimized CNN.
Despite choosing a priori the number of layers, kernel size, filters, strides and all the other possible hyperparameters that could be set when training a CNN, using the keras tuner library the best combination of them could be chosen automatically. The method used to find the best hyperparameters is Bayesian Optimization, a strategy that creates a probabilistic model of the objective function and then exploits this model to select the most promising hyperparameters for evaluation. This works quite efficiently as it uses past trials to make better decisions about future ones. (*Medium*, n.d.)
Despite the initial input layer, the structure of the CNN resulting from the hyperparameters optimization is :

- Convolution Layer #1:
  - Kernel size: (3,3)
  - Strides: (1,1)
  - Number of output filters: 64
  - Padding: None

- Activation function: ReLU
- Regularization L2

- Pooling Layer #1:
  - Pooling type: max
  - Pool Size: (2,2)
  - Strides: (2,2)
  - Padding: None
- Dropout Layer #1:
  - Dropout rate: 20%
- Convolutional Layer #2:
  - Kernel Size: (3, 3)
  - Strides: (1, 1)
  - Filters number of output filters: 96
  - Padding: None
  - Activation: Relu
  - Regularization L2
- Pooling Layer #2:
  - Pooling type: max
  - Pool Size: (2, 2)
  - Strides: (2, 2)
  - Padding: None
- Dropout Layer #2:
  - Dropout rate: 20%
- Flattening Layer
- Fully connected dense layer:
  - Number of Neurons: 512
  - Activation layer: ReLU
- Fully connected output layer:
  - Number of neurons: 1
  - Activation function: Sigmoid

The code used is available at appendix VII.

7. EVALUATION

After building and training the three models as described in the previous paragraphs, the next step was to evaluate them. During the evaluation phase, the models, with both frozen layers and the last 10 layers unfrozen, were assessed based on their loss value, accuracy, F1 score, precision, and recall.

7.1. Metrics

Before showing the results obtained for each model, let's explore how do these metrics are calculated:

- **Loss value**: it is the value assumed by the chosen loss function after an epoch is performed. In this case the function used was Binary Cross-Entropy with Logits Loss, a combination of the sigmoid function and the binary cross entropy loss BCEWithLogits(x,y) = $\frac{1}{N}\sum_{i=1}^{N}(\max(x_i, 0) - x_i\, y_i + \log(1 + e^{-|x_i|}))$
- **Accuracy**: $\frac{True\ Positives\ (TP)+True\ Negatives\ (TN)}{True\ Positives(TP)+True\ Negatoves(TN)+False\ Positives(FP)+False\ Negatives(FN)}$
- **Precision**: $\frac{TP}{TP+FP}$
- **Recall**: $\frac{TP}{TP+FN}$
- **F1**: $\frac{2*Precision*Recall}{Precision+Recall}$

7.2. Results

The results obtained after the training phase are the following:

7.2.1. ResNet50

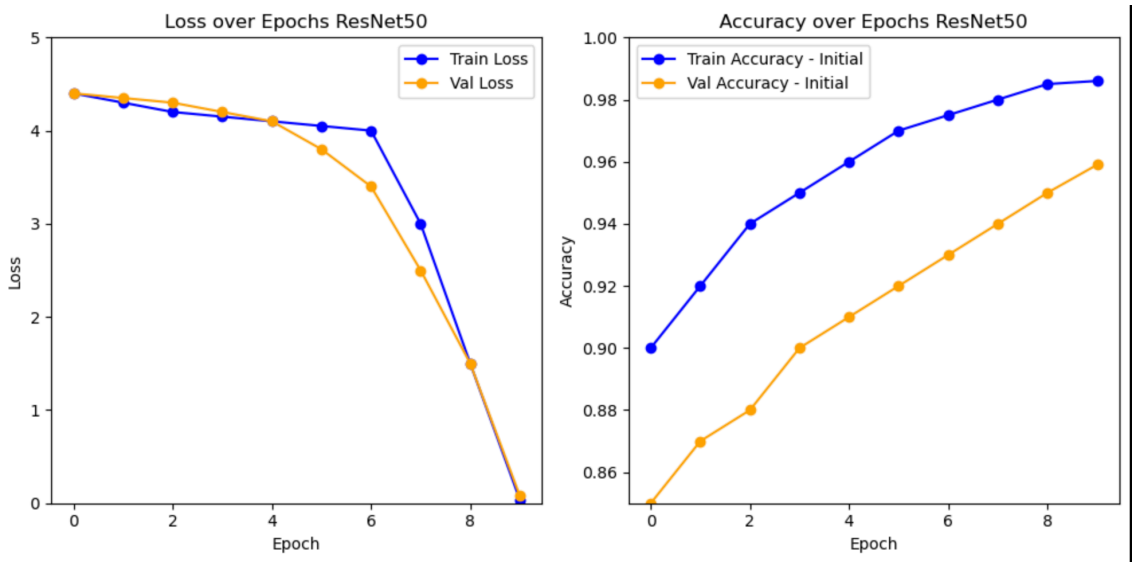


Image 7.2.1.1

The image 7.2.1.1 above shows the evolution of the loss and the accuracy score of the training and validation set over the epochs. The gait of the loss score is quite similar between the two, and it can also be seen how the value of the loss score starts to drop from a certain point onwards, meaning that the model starts to generalize well and it learns effectively. On the other side, while the training set scores consistently more than the validation set over the epochs, they both increase steadily.

ResNet 50 Confusion Matrix						
	Training Set		Validation Set		Test Set	
	PN	PP	PN	PP	PN	PP
AN	6711	120	1134	53	205	29
AP	72	6792	70	1743	10	380

Table 7.2.1.1

ResNet 50			
	Training Set	Validation Set	Test Set
Loss Score	0.038	0.0796	0.09
Accuracy	0.986	0.959	0.938
Precision	0.982	0.992	0.929
Recall	0.989	0.983	0.974
F1	0.985	0.988	0.951

Table 7.2.1.2

These two tables (table 7.2.1.1 and table 7.2.1.2) show the metrics of the ResNet50 model. In the first table it can be seen the confusion matrix of the model in each set of data; a shared pattern is the propensity of the model to predict more false positives (the upper right cell) than false negative (bottom left cell). The second table shows loss score, accuracy, precision, recall and F1 score. The results on the training are very good while they start to decrease as long as we move to the test set. All the metrics of the test set are very solid from the accuracy to the F1 but the one that seems to behave more similarly to the other datasets is the recall; in the test set it scores 97%, a very satisfying result making the model reliable for the task we need to fulfill.

7.2.2. VGG16

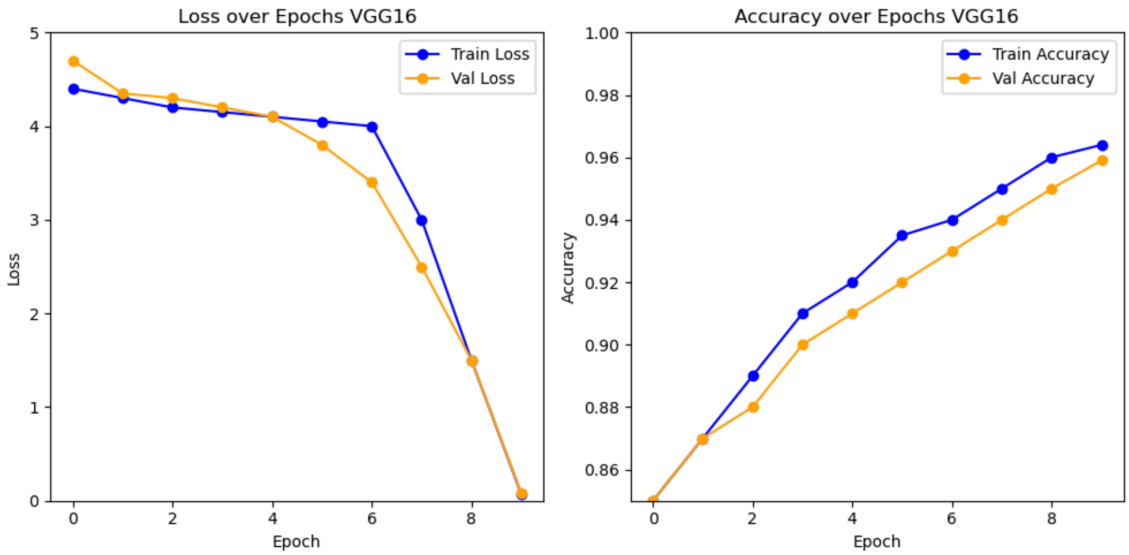


Image 7.2.2.1

What have been previously said for the ResNet50 can be considered valid also in this case; the metrics in the training set are slightly better than the ones in the validation set but also in this case, in both graphs can be noticed a similar trend of the metrics. What changes with respect to the previous case is the fact that looking at the accuracy graphs it can be see how the training set and the validation set performs likewise along the epochs.

VGG16 Confusion Matrix						
	Training Set		Validation Set		Test Set	
	PN	PP	PN	PP	PN	PP
AN	6635	238	1178	9	188	46
AP	257	6599	113	1700	2	388

Table 7.2.2.1

VGG16			
	Training Set	Validation Set	Test Set
Loss Score	0.07	0.0796	0.15
Accuracy	0.964	0.959	0.923
Precision	0.965	0.995	0.89
Recall	0.961	0.937	0.994
F1	0.96	0.97	0.94

Table 7.2.2.2

Table 7.2.2.1 shows the confusion matrix of the VGG16 model; in this case the model seems to be more prone to predict false negatives, a disadvantage for our task, decreasing dramatically this number in the test set.

As for the ResNet model, also for VGG16 the performances generally decline as we move to the test set. The VGG16 model achieves a strong recall and accuracy on the test set while precision is less satisfying at 89. This indicates that out of all the patients predicted to be ill, 89% are actually ill, while demonstrating strong performances in identifying pneumonia cases, the model successfully identifies nearly all of the people suffering from pneumonia.

7.2.3. Optimized CNN

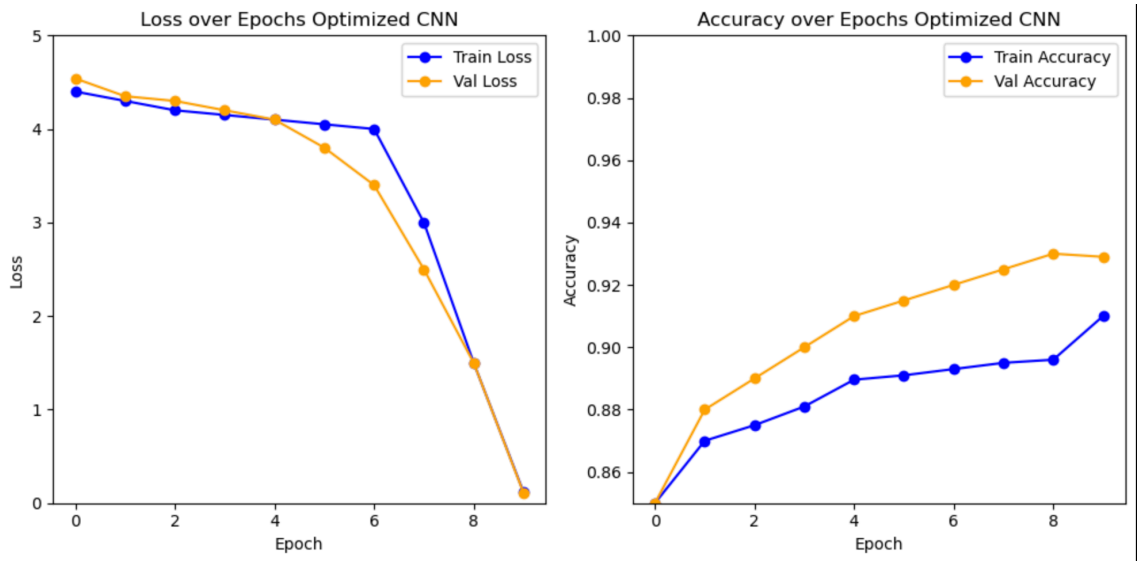


Image 7.2.3.1

As highlighted from image 7.2.3.1 also for the optimized CNN the Loss score behavior over the epochs is quite similar between training and validation set, with validation set scoring a bit better than the training set at the end of the 10th training epoch; on the right side of the image, differently from the other two cases, it is clear how the accuracy of the training phase is overperformed by the accuracy of the validation set.

Optimized CNN Confusion Matrix						
	Training Set		Validation Set		Test Set	
	PN	PP	PN	PP	PN	PP
AN	5748	264	1177	117	134	100
AP	444	6382	95	1611	27	363

Table 7.2.3.

Optimized CNN			
	Training Set	Validation Set	Test Set
Loss Score	0.118	0.11	0.544
Accuracy	0.91	0.929	0.796
Precision	0.96	0.93	0.794
Recall	0.935	0.94	0.93
F1	0.947	0.939	0.85

Table 7.2.3.2

The table 7.2.3.1 shows a model that in the training set is more inclined to false negatives than on false positives, but this trend changes in the other two sets of data where the false positives are more than the false negatives. Despite the latter feature, that can be considered as a good feature of the model, it doesn't have good enough results; the accuracy and the precision in the test set are even less than 80% while the recall is quite good, as shown in table 7.2.3.2.

Despite the positive score on the recall the other results are too poor to challenge the other two models in the final comparison.

All three models were comprehensively evaluated over the datasets, and it was found that the performances for all of them, ResNet50, VGG16, and the optimized CNN, were highly satisfactory over the tested scores. Each model reported above 90% in almost all the metrics, with a few only falling slightly short of the 90% mark (other than some metrics of the optimized CNN that fell below 80%). These would therefore suggest that all the trained models performed very well in addressing the research questions put forth in the study.

Despite all models demonstrated to be reliable, only one of them will be chosen as the best one, to do so we will be focusing on how much each model scored on the test set.

This is because the test set represents realistic conditions under which the model would actually be deployed, a situation where the model does not know the real classification of

an X-ray scan in advance, this shows that the model will perform realistically and practically.

As it has been explained for the three datasets, also for the metrics it will be needed to choose one of them. Despite Accuracy and F1 usually describe how generally a model performs, in this case they are not the most suited metrics.
It is important for our model to minimize the number of FN (False Negatives), the patients that are suffering from pneumonia but are not caught by the model and are labeled as healthy. Labeling a healthy patient as ill will result in him undergoing unnecessary additional examinations, which will eventually confirm he is healthy; in contrast, labeling a sick patient as healthy will result in a lack of timely medical intervention, potentially worsening their condition and leading to severe health complications or even death. The metric we should be focused on is then the recall score that tells us how many actual ill patients are caught by the model.
Keeping in mind these considerations, ResNet scores 97% recall, VGG16 scores 99% recall and the optimized CNN scores 93%.
Choosing the most suited model depends on the task that needed to be performed. Of course if Optimized CNN had scored more than the other two in Recall it would have not been chosen as the best one because the accuracy gap to the other two is too big; minimizing FN is the primary goal of this project but it would be not efficient neither to have too many FP (false positive) because it could result in making patient spending more money and being unnecessarily scared about their condition and especially, it would result in hospital overcrowding making more difficult to those who really need to have access to medical care. VGG16 could sound as the best choice but looking at the confusion matrices of the model, the VGG16 model in the training and validation set was more prone to have more false negatives that false positives and only in the test set the trend changes; this is a ringing bell of the reliability of the model. Differently the ResNet50 model is more coherent all along the sets scoring a solid 97% on the test set in the recall metric together with a very high accuracy, F1 score and precision.
That why the ResNet50 is the best model to accomplish the task.

7.3. Grad-Cam

All along this research we have been talking about machine learning models, their structure and their ability to distinguish those people that are suffering from pneumonia from those one that are not, but despite telling which parameters each model was set to, it has been never discussed in real detail how the each model assigns an X-Ray image to a category or to the other.

CNN models are often seen as black boxes since the process that they follow in segmenting images is not very clear, people just take the output of the model as reliable; in describing the structure of a common CNN we have debated about filters or kernel whose role is the one of catching the features that are fundamental in performing the classification task like shapes, edges, corners and so on. This operation is performed each time a convolution layer is put in the structure of the CNN and depending on the depth of the network at which each layer is positioned, the filters will have to detect different features. To make all this process clearer to people exploiting ML tools it has been developed an ad -hoc library called Gradient Weighted Class Activation Mapping, more commonly known as Grad-Cam. It uses the gradients of any target class flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the image for predicting the concept (Reiff, 2022).

The output of Grad-Cam is a image that highlights in red the zones of the input image that are considered the most in preforming the classification task and in blue the zones that are considered the least.

Considering convolutional layers at different depths of the network will result in different Grad-Cam output images, the more we go toward the final layer, the more the red highlights will be actual zones the more we go toward the initial convolutional layers the more the highlights will be low level features (Reiff, 2022).

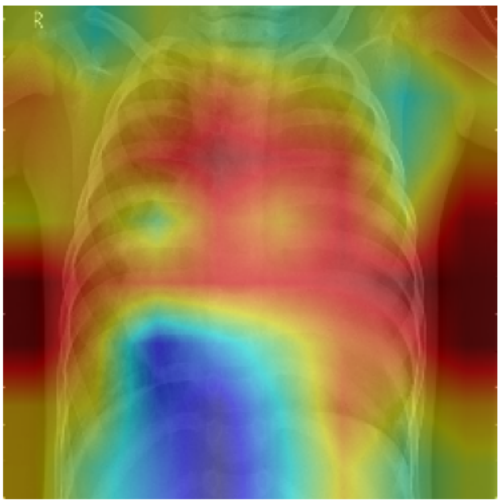


Image 7.3.1

This is an example of a Grad-Cam image generated by the ResNet50 model. How it could have been imagined, it is clear how the red zones corresponds to the position where human lungs are since pneumonia affects that part of the body, while all the other parts of the X-Ray have lower level of importance in the classification.

8. CONCLUSIONS

In this work, it has been explored the change that Machine Learning is bringing in the healthcare field, particularly in diagnostics, predictive analytics, and radiology. This huge volume of data can help health providers make more informed decisions exploiting the strength of nowadays robust algorithms. Specifically, in the subject that has been analyzed, CNN-based models have been demonstrated to have very high performance, mostly over 90% accuracy, including ResNet50 and VGG16. Such models could be included in routine diagnostics; meanwhile, provide consistent, objective, and scalable assessments which are not normally possible with traditional methods that are based on human interpretation. This will include increased sensitivity in performing pattern analysis at the earliest steps of diseases and thus the possibility of earlier interventions with an increased ability to improve prognosis and decrease worsening.
Besides diagnosis, machine learning has been noted to push forward the frontiers of personalized medicine. The result is more accurate in individualized treatments framed through the analysis of data such as genomic and other patient-specific details. This technique ensures improved performance with fewer side effects, especially in cases like pharmacogenomics that aid in predicting patients' response to drugs.

The final aim of this research, however, was to determine the best model amongst the ones trained. When ResNet50, VGG16, and an optimized CNN models were measured, they scored pretty well in all the metrics analyzed, ensuring their reliability in pneumonia detection. However, ResNet50 managed to achieve a higher reliability in the metrics scored along the analyzed sets compared with all other architectures and hence was better suited for the specific application studied.

In general, while this is a field with tremendous potential for improving health, the dominant current issues are the ethical concerns like especially data privacy, algorithmic bias, job loss, and the erosion of the human touch in patient care. However, it is a very optimistic field, particularly with models that keep getting better and adjusting to new data. As shown in the case of the COVID-19 pandemic, machine learning can serve as one of the pivotal players in responding to outbreaks of emergent health issues.

To conclude, clinical decision-making is improved through AI-enabled tools that also aid in population health management to benefit patients by overcoming critical challenges such as shortages of specialized personnel and limited diagnostic resources. More importantly, practices like task shifting, now facilitated by AI, are proving effective enough to help maintain quality care. It has also been stated that the ability of portable devices to host ML-based diagnostic tools will further help in empowering health systems with appropriate, accurate medical evaluation, linked to positive general patient outcomes in such regions.

The integration of AI and ML technologies in the healthcare sector, bears a lot of promise for improving medical care both in developed countries and in LMICs countries. In LMICs countries, it has the potential to innovate the way in which healthcare services can be provided within low-resource areas by making advanced diagnostic tools accessible and affordable, resulting in the worthiest possible usage of AI; in developed countries, the benefits may not be centered around improving diagnostic quality or accessibility but rather on gaining deeper insights into diseases and their progression. This knowledge enables physicians to prescribe more targeted tests, ultimately reducing healthcare costs bearing on the state coffers providing economic benefits to the countries as a whole.

CITATIONS

1. Absar, N., Mamur, B., Mahmud, A., Emran, T.B., Khandaker, M.U., Faruque, M.R.I., Osman, H., Elzaki, A. and Elkhader, B.A., 2022. Development of a computer-aided tool for detection of COVID-19 pneumonia from CXR images using machine learning algorithm. Journal of Radiation Research and Applied Sciences.
2. Ahmad, M.A., Eckert, C. and Teredesai, A., 2018, August. Interpretable machine learning in healthcare. In Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics .
3. Ahmad, M.A., Patel, A., Eckert, C., Kumar, V. and Teredesai, A., 2020, August. Fairness in machine learning for healthcare. In Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining.
4. Al Mamlook, R.E., Chen, S. and Bzizi, H.F., 2020, July. Investigation of the performance of machine learning classifiers for pneumonia detection in chest X-ray images. In 2020 IEEE International Conference on Electro Information Technology (EIT).
5. Alanazi, A., 2022. Using machine learning for healthcare challenges and opportunities. Informatics in Medicine Unlocked.
6. Allgaier, J., Mulansky, L., Draelos, R.L. and Pryss, R., 2023. How does the model make predictions? A systematic literature review on the explainability power of machine learning in healthcare. Artificial Intelligence in Medicine.
7. An, Q., Rahman, S., Zhou, J. and Kang, J.J., 2023. A comprehensive review on machine learning in healthcare industry: classification, restrictions, opportunities and challenges. Sensors.
8. Arvindhan, M., Rajeshkumar, D. and Pal, A.L., 2021. A review of challenges and opportunities in machine learning for healthcare. Exploratory Data Analytics for Healthcare.
9. Ballamudi, V.K.R., 2016. Utilization of Machine Learning in a Responsible Manner in the Healthcare Sector. Malaysian Journal of Medical and Biological Research.
10. Barakat, N., Awad, M. and Abu-Nabah, B.A., 2023. A machine learning approach on chest X-rays for pediatric pneumonia detection. Digital Health.
11. Barhoom, A.M. and Abu-Naser, S.S., 2022. Diagnosis of pneumonia using deep learning.
12. Bhardwaj, R., Nambiar, A.R. and Dutta, D., 2017, July. A study of machine learning in healthcare. In 2017 IEEE 41st annual computer software and applications conference (COMPSAC).

13. Callahan, A. and Shah, N.H., 2017. Machine learning in healthcare. In Key advances in clinical informatics.
14. Chandra, T.B. and Verma, K., 2020. Pneumonia detection on chest x-ray using machine learning paradigm. In Proceedings of 3rd International Conference on Computer Vision and Image Processing: CVIP 2018.
15. Char, D.S., Shah, N.H. and Magnus, D., 2018. Implementing machine learning in health care—addressing ethical challenges. New England Journal of Medicine.
16. Cheekuri, S.V., Veeramachaneni, M., Manikandan, V.M. and Kumar, R.K., 2024, May. A Machine Learning-Based Pneumonia Detection System. In 2024 5th International Conference for Emerging Technology.
17. Chen, I.Y., Joshi, S., Ghassemi, M. and Ranganath, R., 2021. Probabilistic machine learning for healthcare. Annual review of biomedical data science.
18. Chen, M., Hao, Y., Hwang, K., Wang, L. and Wang, L., 2017. Disease prediction by machine learning over big data from healthcare communities. Ieee Access.
19. Chua, M., Kim, D., Choi, J., Lee, N.G., Deshpande, V., Schwab, J., Lev, M.H., Gonzalez, R.G., Gee, M.S. and Do, S., 2023. Tackling prediction uncertainty in machine learning for healthcare. Nature Biomedical Engineerin.
20. Das, S., Pradhan, S.K., Mishra, S., Pradhan, S. and Pattnaik, P.K., 2022, March. A machine learning based approach for detection of pneumonia by analyzing chest X-ray images. In 2022 9th International Conference on Computing for Sustainable Global Development.
21. Erdem, E. and Aydin, T., 2021. Detection of Pneumonia with a Novel CNN-based Approach. Sakarya University Journal of Computer and Information Sciences.
22. Feng, Q., Du, M., Zou, N. and Hu, X., 2022. Fair machine learning in healthcare: A review.
23. Gabriel, O.T., 2023. Data privacy and ethical issues in collecting health care data using artificial intelligence among health workers (Master's thesis, Center for Bioethics and Research).
24. Gabruseva, T., Poplavskiy, D. and Kalinin, A., 2020. Deep learning for automatic pneumonia detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops .
25. Gichoya, J.W., McCoy, L.G., Celi, L.A. and Ghassemi, M., 2021. Equity in essence: a call for operationalising fairness in machine learning for healthcare. BMJ health & care informatics, 28(1).
26. GM, H., Gourisaria, M.K., Rautaray, S.S. and Pandey, M.A.N.J.U.S.H.A., 2021. Pneumonia detection using CNN through chest X-ray. Journal of Engineering Science and Technology (JESTEC.

27. Goyal, S. and Singh, R., 2023. Detection and classification of lung diseases for pneumonia and Covid-19 using machine and deep learning techniques. Journal of Ambient Intelligence and Humanized Computing.
28. Gupta, S. and Sedamkar, R.R., 2020. Machine learning for healthcare: Introduction. Machine learning with health care perspective: Machine learning and healthcare.
29. Habehh, H. and Gohel, S., 2021. Machine learning in healthcare. Current genomics.
30. Jadhav, S., Kasar, R., Lade, N., Patil, M. and Kolte, S., 2019. Disease prediction by machine learning from healthcare communities. International Journal of Scientific Research in Science and Technology.
31. Jain, D.K., Singh, T., Saurabh, P., Bisen, D., Sahu, N., Mishra, J. and Rahman, H., 2022. Deep Learning-Aided Automated Pneumonia Detection and Classification Using CXR Scans. Computational intelligence and neuroscience, 2022(1).
32. Javaid, M., Haleem, A., Singh, R.P., Suman, R. and Rab, S., 2022. Significance of machine learning in healthcare: Features, pillars and applications. International Journal of Intelligent Networks.
33. Jia, Y., McDermid, J., Lawton, T. and Habli, I., 2022. The role of explainability in assuring safety of machine learning in healthcare. IEEE Transactions on Emerging Topics in Computing.
34. Kareem, A., Liu, H. and Sant, P., 2022. Review on pneumonia image detection: A machine learning approach. Human-Centric Intelligent Systems.
35. Kasula, B.Y., 2021. Machine Learning in Healthcare: Revolutionizing Disease Diagnosis and Treatment. International Journal of Creative Research In Computer Technology and Design.
36. Kelly, C.J., Karthikesalingam, A., Suleyman, M., Corrado, G. and King, D., 2019. Key challenges for delivering clinical impact with artificial intelligence. BMC medicine.
37. Khan, B., Fatima, H., Qureshi, A., Kumar, S., Hanan, A., Hussain, J. and Abdullah, S., 2023. Drawbacks of artificial intelligence and their potential solutions in the healthcare sector. Biomedical Materials & Devices.
38. Kundu, R., Das, R., Geem, Z.W., Han, G.T. and Sarkar, R., 2021. Pneumonia detection in chest X-ray images using an ensemble of deep learning models. PloS one.
39. Leimanis, A. and Palkova, K., 2021. Ethical guidelines for artificial intelligence in healthcare from the sustainable development perspective. European Journal of Sustainable Development.
40. Liu, C., Wang, X., Liu, C., Sun, Q. and Peng, W., 2020. Differentiating novel coronavirus pneumonia from general pneumonia based on machine learning. Biomedical engineering online.

41. MacKay, C., Klement, W., Vanberkel, P., Lamond, N., Urquhart, R. and Rigby, M., 2023. A framework for implementing machine learning in healthcare based on the concepts of preconditions and postconditions. Healthcare Analytics.
42. McCoy, L.G., Banja, J.D., Ghassemi, M. and Celi, L.A., 2020. Ensuring machine learning for healthcare works for all. BMJ Health & Care Informatics.
43. Mohanty, S.N., Nalinipriya, G., Jena, O.P. and Sarkar, A. eds., 2021. Machine learning for healthcare applications. John Wiley & Sons.
44. Muhammad, Y., Alshehri, M.D., Alenazy, W.M., Vinh Hoang, T. and Alturki, R., 2021. Identification of pneumonia disease applying an intelligent computational framework based on deep learning and machine learning techniques. Mobile Information Systems, 2021(1).
45. Mustafa, A. and Rahimi Azghadi, M., 2021. Automated machine learning for healthcare and clinical notes analysis. Computers.
46. Naik, N., Hameed, B.Z., Shetty, D.K., Swain, D., Shah, M., Paul, R., Aggarwal, K., Ibrahim, S., Patil, V., Smriti, K. and Shetty, S., 2022. Legal and ethical consideration in artificial intelligence in healthcare: who takes responsibility?. Frontiers in surgery.
47. Naresh, V.S. and Thamarai, M., 2023. Privacy-preserving data mining and machine learning in healthcare: Applications, challenges, and solutions. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery.
48. Nayyar, A., Gadhavi, L. and Zaman, N., 2021. Machine learning in healthcare: review, opportunities and challenges. Machine Learning and the Internet of Medical Things in Healthcare.
49. Naz, A., Ghous, H. and Khan, N., 2020. Literature review on X-ray based pneumonia detection using machine learning and deep learning methods. LC International Journal of .
50. Ngiam, K.Y. and Khor, W., 2019. Big data and machine learning algorithms for health-care delivery. The Lancet Oncology.
51. Nimbolkar, S., Thakare, A., Mitra, S., Biranje, O. and Sutar, A., 2022. Detection of Pneumonia Using Machine Learning and Deep Learning Techniques: An Analytical Study. Object Detection by Stereo Vision Images.
52. Panesar, A., 2019. Machine learning and AI for healthcare
53. Pankratz, D.G., Choi, Y., Imtiaz, U., Fedorowicz, G.M., Anderson, J.D., Colby, T.V., Myers, J.L., Lynch, D.A., Brown, K.K., Flaherty, K.R. and Steele, M.P., 2017. Usual interstitial pneumonia can be detected in transbronchial biopsies using machine learning. Annals of the American Thoracic Society.
54. Pap, K. and Hrnčić, M., 2019. Implementation of intelligent model for pneumonia detection. Tehnički glasnik.

55. Pattnayak, P. and Panda, A.R., 2021. Innovation on machine learning in healthcare services—An introduction. Technical Advancements of Machine Learning in Healthcare.
56. Prakash, S., Balaji, J.N., Joshi, A. and Surapaneni, K.M., 2022. Ethical Conundrums in the application of artificial intelligence (AI) in healthcare—a scoping review of reviews. Journal of Personalized Medicine.
57. Qayyum, A., Qadir, J., Bilal, M. and Al-Fuqaha, A., 2020. Secure and robust machine learning for healthcare: A survey. IEEE Reviews in Biomedical Engineering.
58. Qu, Y., Meng, Y., Fan, H. and Xu, R.X., 2022. Low-cost thermal imaging with machine learning for non-invasive diagnosis and therapeutic monitoring of pneumonia. Infrared Physics & Technology.
59. Račić, L., Popović, T. and Šandi, S., 2021, February. Pneumonia detection using deep learning based on convolutional neural network. In 2021 25th International Conference on Information Technology (IT).
60. Rahman, T., Chowdhury, M.E., Khandakar, A., Islam, K.R., Islam, K.F., Mahbub, Z.B., Kadir, M.A. and Kashem, S., 2020. Transfer learning with deep convolutional neural network (CNN) for pneumonia detection using chest X-ray. Applied Sciences.
61. Rani, S., Pareek, P.K., Kaur, J., Chauhan, M. and Bhambri, P., 2023, February. Quantum machine learning in healthcare: Developments and challenges. In 2023 IEEE International Conference on Integrated Circuits and Communication Systems.
62. Rasheed, K., Qayyum, A., Ghaly, M., Al-Fuqaha, A., Razi, A. and Qadir, J., 2022. Explainable, trustworthy, and ethical machine learning for healthcare: A survey. Computers in Biology and Medicine.
63. Rastogi, M., Vijarania, D.M. and Goel, D.N., 2022, August. Role of machine learning in healthcare sector. In Proceedings of the International Conference on Innovative Computing & Communication (ICICC).
64. Sabry, F., Eltaras, T., Labda, W., Alzoubi, K. and Malluhi, Q., 2022. Machine learning for healthcare wearable devices: the big picture. Journal of Healthcare Engineering, 2022(1).
65. Saleem, T.J. and Chishti, M.A., 2020. Exploring the applications of machine learning in healthcare. International Journal of Sensors Wireless Communications and Control.
66. Sanchez, P., Voisey, J.P., Xia, T., Watson, H.I., O'Neil, A.Q. and Tsaftaris, S.A., 2022. Causal machine learning for healthcare and precision medicine. Royal Society Open Science.
67. Shailaja, K., Seetharamulu, B. and Jabbar, M.A., 2018, March. Machine learning in healthcare: A review. In 2018 Second international conference on electronics, communication and aerospace .

68. Siddiq, M., 2021. Integration of Machine Learning in Clinical Decision Support Systems. Eduvest-Journal of Universal Studies.
69. Siddique, S. and Chow, J.C., 2021. Machine learning in healthcare communication.
70. Stokes, K., Castaldo, R., Franzese, M., Salvatore, M., Fico, G., Pokvic, L.G., Badnjevic, A. and Pecchia, L., 2021. A machine learning model for supporting symptom-based referral and diagnosis of bronchitis and pneumonia in limited resource settings. Biocybernetics and biomedical engineering.
71. Swain, S., Bhushan, B., Dhiman, G. and Viriyasitavat, W., 2022. Appositeness of optimized and reliable machine learning for healthcare: a survey. Archives of Computational Methods in Engineering.
72. Swetha, K.R., Niranjanamurthy, M., Amulya, M.P. and Manu, Y.M., 2021, July. Prediction of pneumonia using big data, deep learning and machine learning techniques. In 2021 6th International Conference on Communication and Electronics Systems.
73. Tilve, A., Nayak, S., Vernekar, S., Turi, D., Shetgaonkar, P.R. and Aswale, S., 2020, February. Pneumonia detection using deep learning approaches. In 2020 international conference on emerging trends in information technology and engineering.
74. Toğaçar, M., Ergen, B., Cömert, Z. and Özyurt, F., 2020. A deep feature learning model for pneumonia detection applying a combination of mRMR feature selection and machine learning models. Irbm.
75. Toh, C. and Brody, J.P., 2021. Applications of machine learning in healthcare. Smart manufacturing: When artificial intelligence meets the internet of things.
76. Tsai, M.J. and Tao, Y.H., 2019, December. Machine learning based common radiologist-level pneumonia detection on chest X-rays. In 2019 13th International Conference on Signal Processing and Communication Systems .
77. Usmani, U.A. and Jaafar, J., 2022, November. Machine learning in healthcare: current trends and the future. In International Conference on Artificial Intelligence for Smart Community: AISC 2020, 17–18 December, Universiti Teknologi Petronas, Malaysia Singapore: Springer Nature Singapore.
78. Varshni, D., Thakral, K., Agarwal, L., Nijhawan, R. and Mittal, A., 2019, February. Pneumonia detection using CNN based feature extraction. In 2019 IEEE international conference on electrical, computer and communication technologies.
79. Wan, J., 2022. "Artificial Intelligence, Machine Learning and Deep Learning–Limitations: Privacy and Data Security Issues" Chapter. In Artificial Intelligence in Medicine: Applications, Limitations and Future Directions Singapore: Springer Nature Singapore.
80. Yahyaoui, A. and Yumuşak, N., 2021, September. Deep and machine learning towards pneumonia and asthma detection. In 2021 International Conference

on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT) .
81. Yaseliani, M., Hamadani, A.Z., Maghsoodi, A.I. and Mosavi, A., 2022. Pneumonia detection proposing a hybrid deep convolutional neural network based on two parallel visual geometry group architectures and machine learning classifiers. IEEE access.
82. Yee, S.L.K. and Raymond, W.J.K., 2020, September. Pneumonia diagnosis using chest X-ray images and machine learning. In proceedings of the 2020 10th international conference on biomedical engineering and technology .
83. Zhang, A., Xing, L., Zou, J. and Wu, J.C., 2022. Shifting machine learning for healthcare from development to deployment and from models to data. Nature Biomedical Engineering.
84. Chollet, B. F. (n.d.). Building powerful image classification models using very little data.

85. *imbalanced classification*. (2022).
86. Academy, E. (2023, August 13). Why is it important to split the data into training and validation sets? How much data is typically allocated for validation? - EITCA Academy. EITCA Academy.
87. Educative. (n.d.). What is PyTorch DataLoader?
88. Ku, C. (2018, November 28). How does DataLoader work in PyTorch? - Noumena - Medium. *Medium*.
89. *PyTorch DataLoader: Features, Benefits, and How to Use it | Saturn Cloud Blog*. (2023, October 4).
90. Mandal, M. (2024, June 7). *Introduction to Convolutional Neural Networks (CNN)*. Analytics Vidhya.
91. Chatterjee, C. C. (2021, December 11). Basics of the classic CNN - towards data science. *Medium*.
92. Mishra, M. (2021, December 15). Convolutional neural networks, explained - towards data science. *Medium*.
93. Chaudhary, V. (2023, May 19). *What is "padding" in Convolutional Neural Network?* AlmaBetter.
94. NeuralNet, M. (n.d.). *Calculating the output size of convolutions and transpose convolutions*.
95. *A gentle introduction to pooling layers for convolutional neural networks*. (2019).
96. Ali, M. S. (2022, June 28). Flattening CNN layers for Neural Network and basic concepts. *Medium*.
97. GeeksforGeeks. (2024, July 16). *Activation functions in Neural Networks*. GeeksforGeeks

98. Jadon, S. (2022, February 3). Introduction to different activation functions for deep learning. *Medium*.
99. Rehman, H. U. (2024, February 5). Binary Image Classification using Resnet50 - Habib Ur Rehman - Medium. *Medium*.
100. Danielsen, N. (2021, December 12). Simple Image Classification with ResNet-50 - Nina Danielsen - Medium. *Medium*.
101. Kien, C. T. (2022, October 8). Skip connection and explanation of ResNet - Chau Tuan Kien - medium. *Medium*.
102. Reshan MSA, Gill KS, Anand V, Gupta S, Alshahrani H, Sulaiman A, Shaikh A. Detection of Pneumonia from Chest X-ray Images Utilizing MobileNet Model. Healthcare (Basel). 2023 May 26;11(11):1561. doi: 10.3390/healthcare11111561. PMID: 37297701; PMCID: PMC10252226.
103. Cannata, G. P. (2021, December 16). Additive skip connection - Giuseppe Pio Cannata - Medium. *Medium*.
104. Great Learning. (2022, January 5). Everything you need to know about VGG16 - Great Learning - Medium. *Medium*.
105. Thakur, R. (2024, January 9). Step by step VGG16 implementation in Keras for beginners. *Medium*. https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c
106. Makarenko, A. (2023, March 26). Boost Your Image Classification Model with pretrained VGG-16. *Medium*.
107. Dutta, S. (2024, April 23). Hyperparameter Tuning with Keras Tuner and TensorFlow. *Medium*.
108. Reiff, D. (2022, May 12). Understand your Algorithm with Grad-CAM - Towards Data Science. *Medium*. https://towardsdatascience.com/understand-your-algorithm-with-grad-cam-d3b62fce353
109. Eswaran, U., & Khang, A. (2024). Artificial intelligence (AI)-Aided Computer Vision (CV) in healthcare system. In *CRC Press eBooks* (pp. 125–137). https://doi.org/10.1201/9781003429609-8
110. Ali, O., Abdelbaki, W., Shrestha, A., Elbasi, E., Alryalat, M. a. A., & Dwivedi, Y. K. (2023). A systematic literature review of artificial intelligence in the healthcare sector: Benefits, challenges, methodologies, and functionalities. *Journal of Innovation & Knowledge*, *8*(1), 100333. https://doi.org/10.1016/j.jik.2023.100333
111. Reshan, M. S. A., Gill, K. S., Anand, V., Gupta, S., Alshahrani, H., Sulaiman, A., & Shaikh, A. (2023). Detection of Pneumonia from Chest X-ray Images Utilizing MobileNet Model. *Healthcare*, *11*(11), 1561. https://doi.org/10.3390/healthcare11111561
112. Amisha, Malik, P., Pathania, M., & Rathaur, V. K. (2019). Overview of artificial intelligence in medicine. *Journal of Family Medicine and Primary Care*, *8*(7), 2328. https://doi.org/10.4103/jfmpc.jfmpc_440_19

113. Yadav, H. (2023, May 31). Dropout in neural networks - towards data science. *Medium*. https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9
114. Doshi, K. (2022, January 6). Batch Norm Explained Visually — How it works, and why neural networks need it. *Medium*. https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739
115. *What is pneumonia? | NHLBI, NIH*. (2022, March 24). NHLBI, NIH. https://www.nhlbi.nih.gov/health/pneumonia
116. Vishal. (2024, January 29). Transfer learning in Convolution Neural Network - Vishal - Medium. *Medium*. https://medium.com/@vishal025/transfer-learning-in-convolution-neural-network-b69504f1d052
117. *Chest X-Ray images (Pneumonia)*. (2018, March 24). https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia
118. *Convolutional Neural Networks (CNN): Step 4 - Full connection - Blogs - SuperDataScience | Machine Learning | AI | Data Science Career | Analytics | Success*. (n.d.). SuperDataScience. https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection
119. Rajpurkar, P., & Lungren, M. P. (2023). The current and future state of AI interpretation of medical images. *New England Journal of Medicine*, *388*(21), 1981–1990. https://doi.org/10.1056/nejmra2301725
120. Gaire, C. (2023, July 6). Santé: comment l'IA est devenue l'outil indispensable des radiologues. *BFM BUSINESS*. https://www.bfmtv.com/economie/sante-comment-l-ia-est-devenue-l-outil-indispensable-des-radiologues_AN-202307060056.html
121. Szepesi, P., & Szilágyi, L. (2022). Detection of pneumonia using convolutional neural networks and deep learning. *Journal of Applied Biomedicine*, *42*(3), 1012–1022. https://doi.org/10.1016/j.bbe.2022.08.001
122. Kareem, A., Liu, H., & Sant, P. (2022). Review on Pneumonia Image Detection: A Machine Learning Approach. *Human-Centric Intelligent Systems*, *2*(1–2), 31–43. https://doi.org/10.1007/s44230-022-00002-2

APPENDIX

- Appendix I, code for obtaining average size of images

```
import os
from PIL import Image
import numpy as np

def get_image_dimensions(image_path):
    with Image.open(image_path) as img:
        return img.size  # returns (width, height)

def calculate_average_and_sd_dimensions(folder_path):
    widths = []
    heights = []

    for filename in os.listdir(folder_path):
        if filename.lower().endswith(('.png', '.jpg', '.jpeg', '.gif', '.bmp')):
            image_path = os.path.join(folder_path, filename)
            try:
                width, height = get_image_dimensions(image_path)
                widths.append(width)
                heights.append(height)
            except Exception as e:
                print(f"Error processing {filename}: {e}")

    if not widths or not heights:
        print("No images found in the folder.")
        return

    avg_width = np.mean(widths)
    avg_height = np.mean(heights)
    sd_width = np.std(widths)
    sd_height = np.std(heights)

    print(f"Average Width: {avg_width:.2f}")
    print(f"Average Height: {avg_height:.2f}")
    print(f"Width Standard Deviation: {sd_width:.2f}")
    print(f"Height Standard Deviation: {sd_height:.2f}")
```


Appendix II Data augmentation Code

```
logging.basicConfig(level=logging.INFO,                format='%(asctime)s
%(levelname)s:%(message)s')
logger = logging.getLogger()
handler = logging.StreamHandler()
formatter = logging.Formatter('%(asctime)s %(levelname)s:%(message)s')
handler.setFormatter(formatter)
logger.handlers = [handler]  # Replace default handlers with the stream handler

# Set the categories we want to augment
categories = ['NORMAL', 'PNEUMONIA']
categories_dict = {'NORMAL': 6, 'PNEUMONIA': 2}

datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    vertical_flip=True,
    brightness_range=[0.8, 1.2],
    fill_mode='nearest'
)

def process_image(filename, source_directory, final_directory, category):
    img_path = os.path.join(source_directory, filename)
    try:
        img = load_img(img_path)
        x = img_to_array(img)
        x = x.reshape((1,) + x.shape)
        shutil.copy2(img_path, final_directory)
        logger.info(f"Copied original image {filename} to {final_directory}")
```

```
        augment_factor = categories_dict[category]
        for i, batch in enumerate(datagen.flow(x, batch_size=1,
save_to_dir=final_directory, save_prefix='aug', save_format='jpg')):
            if i >= (augment_factor - 1):
                break
        logger.info(f"Generated augmented images for {filename}")
    except Exception as e:
        logger.error(f"Error processing file {filename}: {e}")

for category in categories:
    source_directory =
os.path.join(r"C:\Users\vince\OneDrive\Desktop\Tesi\train", category)
    final_directory =
os.path.join(r"C:\Users\vince\OneDrive\Desktop\Tesi\train\Augmented Images",
category)
    os.makedirs(final_directory, exist_ok=True)

    for filename in os.listdir(source_directory):
        if filename.endswith((".jpg", ".jpeg")):
            thread = Thread(target=process_image, args=(filename, source_directory,
final_directory, category))
            thread.start()
            thread.join(timeout=10)  # 10 seconds timeout for each file

            if thread.is_alive():
                logger.error(f"Timeout error processing file {filename}")
```

- Appendinx III Validation set Creation

```
import os
import shutil

# Define paths
```

```
source_dir = r"C:\Users\vince\OneDrive\Desktop\Tesi\train\Augmented Images"
target_dir = r"C:\Users\vince\OneDrive\Desktop\Tesi\train\Validation Set"

normal_dir = os.path.join(source_dir, "NORMAL")
pneumonia_dir = os.path.join(source_dir, "PNEUMONIA")

# Create target directories if they don't exist
os.makedirs(os.path.join(target_dir, "normal"), exist_ok=True)
os.makedirs(os.path.join(target_dir, "pneumonia"), exist_ok=True)

# Get list of files in each source directory
normal_files = os.listdir(normal_dir)
pneumonia_files = os.listdir(pneumonia_dir)

# Calculate proportions
total_files = len(normal_files) + len(pneumonia_files)
normal_proportion = len(normal_files) / total_files
pneumonia_proportion = len(pneumonia_files) / total_files

# Determine number of files to pick from each directory
num_normal = int(3000 * normal_proportion)
num_pneumonia = 3000 - num_normal

# Randomly select files
selected_normal_files = random.sample(normal_files, num_normal)
selected_pneumonia_files = random.sample(pneumonia_files, num_pneumonia)

# Copy selected files to target directory
for file in selected_normal_files:
    shutil.copy(os.path.join(normal_dir, file), os.path.join(target_dir, "normal", file))

for file in selected_pneumonia_files:
    shutil.copy(os.path.join(pneumonia_dir, file), os.path.join(target_dir, "pneumonia", file))

print(f"Copied {num_normal} normal files and {num_pneumonia} pneumonia files to {target_dir}")
```

```
# Counting the images in each folder
folder_path = r"C:\Users\vince\OneDrive\Desktop\Tesi\train\Augmented Images\NORMAL"

# List all files in the directory
all_files = os.listdir(folder_path)

# Filter out only image files (e.g., jpg, jpeg, png)
image_extensions = ('.jpg', '.jpeg', '.png', '.gif', '.bmp', '.tiff')
image_files = [file for file in all_files if file.lower().endswith(image_extensions)]

# Count the number of image files
num_images = len(image_files)

print(f'Total number of images: {num_images}')
```

- Appendix IV Data Loaders

```
# Load paths
def load_paths(directory, label):
    path_list = []
    for path in os.listdir(directory):
        if path.endswith(".jpeg") or path.endswith(".jpg") or path.endswith(".png"):
            path_list.append((os.path.join(directory, path), label))
    return path_list

train_path_list = load_paths(normal_dir, 0) + load_paths(pneumonia_dir, 1)
val_path_list = load_paths(val_normal_dir, 0) + load_paths(val_pneumonia_dir, 1)
test_path_list = load_paths(test_normal_dir, 0) + load_paths(test_pneumonia_dir, 1)
```

```
# Create DataFrames
train_dataset = pd.DataFrame(train_path_list, columns=['image_path', 'label'])
val_dataset = pd.DataFrame(val_path_list, columns=['image_path', 'label'])
test_dataset = pd.DataFrame(test_path_list, columns=['image_path', 'label'])

# PyTorch Dataset class
class XRayDataset(Dataset):
    def __init__(self, dataframe, transform=None):
        self.dataframe = dataframe
        self.transform = transform

    def __len__(self):
        return len(self.dataframe)

    def __getitem__(self, idx):
        img_path = self.dataframe.iloc[idx, 0]
        label = self.dataframe.iloc[idx, 1]
        try:
            image = Image.open(img_path).convert('L')  # Convert to grayscale
            image = image.convert('RGB')  # Convert grayscale to RGB
        except Exception as e:
            print(f"Error loading image {img_path}: {e}")
            return None, None

        if self.transform:
            image = self.transform(image)

        return image, label

# Create DataLoaders
def collate_fn(batch):
    batch = list(filter(lambda x: x[0] is not None, batch))  # Filter out None images
    return torch.utils.data.dataloader.default_collate(batch)

train_dataset = XRayDataset(train_dataset, transform=transform)
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True, collate_fn=collate_fn)

val_dataset = XRayDataset(val_dataset, transform=transform)
```

```
val_loader = DataLoader(val_dataset, batch_size=64, shuffle=True, collate_fn=collate_fn)

test_dataset = XRayDataset(test_dataset, transform=transform)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False, collate_fn=collate_fn)
```

- Appendix 5 ResNet50

```
# Load the pre-trained ResNet50 model
base_model = models.resnet50(weights=models.ResNet50_Weights.IMAGENET1K_V1)

# Freeze the base model layers
for param in base_model.parameters():
    param.requires_grad = False

# Modify the ResNet50 model
class ModifiedResNet50(nn.Module):
    def __init__(self, base_model):
        super(ModifiedResNet50, self).__init__()
        self.base_model = nn.Sequential(*list(base_model.children())[:-2])
        self.custom_layers = nn.Sequential(
            nn.Conv2d(2048, 512, kernel_size=3, padding=1),
            nn.BatchNorm2d(512),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Dropout(0.5),
            nn.Conv2d(512, 256, kernel_size=3, padding=1),
            nn.BatchNorm2d(256),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Dropout(0.5),
            nn.AdaptiveAvgPool2d((1, 1)),  # Adaptive average pooling to handle variable sizes
```

```
            nn.Flatten()
        )
        self.fc = nn.Sequential(
            nn.Linear(256, 1024),  # Adjusted to match the flattened size
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(1024, 1)
        )

    def forward(self, x):
        x = self.base_model(x)
        x = self.custom_layers(x)
        x = self.fc(x)
        return x

# Create the modified model
model = ModifiedResNet50(base_model)

# Define loss function and optimizer
criterion = nn.BCEWithLogitsLoss()
optimizer = optim.Adam(model.parameters(), lr=0.0001)

# L1 and L2 regularization parameters
l1_lambda = 1e-5
l2_lambda = 1e-4

# Move model to GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Lists to store metrics for initial training phase
train_losses_initial = []
val_losses_initial = []
train_accuracies_initial = []
val_accuracies_initial = []

# Lists to store metrics for fine-tuning phase
train_losses_finetune = []
val_losses_finetune = []
train_accuracies_finetune = []
val_accuracies_finetune = []
```

```
def evaluate(model, data_loader, criterion, device):
    model.eval()
    running_loss = 0.0
    correct = 0
    total = 0
    all_labels = []
    all_predictions = []
    with torch.no_grad():
        for images, labels in data_loader:
            images = images.to(device)
            labels = labels.to(device).float().view(-1, 1)
            outputs = model(images)
            loss = criterion(outputs, labels)
            # L1 and L2 penalties
            l1_penalty = sum(p.abs().sum() for p in model.parameters())
            l2_penalty = sum((p ** 2).sum() for p in model.parameters())
            loss += l1_lambda * l1_penalty + l2_lambda * l2_penalty
            running_loss += loss.item()
            predicted = (torch.sigmoid(outputs) > 0.5).float()
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
            all_labels.extend(labels.cpu().numpy())
            all_predictions.extend(predicted.cpu().numpy())
    cm = confusion_matrix(all_labels, all_predictions)
    f1 = f1_score(all_labels, all_predictions)
    precision = precision_score(all_labels, all_predictions)
    recall = recall_score(all_labels, all_predictions)
    return running_loss / len(data_loader), correct / total, cm, f1, precision,
recall

# Training loop with all layers frozen
num_epochs = 10
for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0

    for images, labels in train_loader:
        images = images.to(device)
        labels = labels.to(device).float().view(-1, 1)
```

```
        # Zero the parameter gradients
        optimizer.zero_grad()

        # Forward pass
        outputs = model(images)
        loss = criterion(outputs, labels)
        # L1 and L2 penalties
        l1_penalty = sum(p.abs().sum() for p in model.parameters())
        l2_penalty = sum((p ** 2).sum() for p in model.parameters())
        loss += l1_lambda * l1_penalty + l2_lambda * l2_penalty

        # Backward pass and optimize
        loss.backward()
        optimizer.step()

        # Print statistics
        running_loss += loss.item()
        predicted = (torch.sigmoid(outputs) > 0.5).float()
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    train_losses_initial.append(running_loss / len(train_loader))
    train_accuracies_initial.append(correct / total)

    val_loss, val_accuracy, _, _, _, _ = evaluate(model, val_loader, criterion,
device)
    val_losses_initial.append(val_loss)
    val_accuracies_initial.append(val_accuracy)

    print(f'Epoch {epoch+1}, Train Loss: {train_losses_initial[-1]}, Train
Accuracy: {train_accuracies_initial[-1]}, Val Loss: {val_losses_initial[-1]},
Val Accuracy: {val_accuracies_initial[-1]}')

# Unfreeze the last 10 layers
for param in list(model.parameters())[-10:]:
    param.requires_grad = True

# Fine-tuning loop
num_finetune_epochs = 5
for epoch in range(num_finetune_epochs):
    model.train()
    running_loss = 0.0
```

```
    correct = 0
    total = 0

    for images, labels in train_loader:
        images = images.to(device)
        labels = labels.to(device).float().view(-1, 1)

        # Zero the parameter gradients
        optimizer.zero_grad()

        # Forward pass
        outputs = model(images)
        loss = criterion(outputs, labels)
        # L1 and L2 penalties
        l1_penalty = sum(p.abs().sum() for p in model.parameters())
        l2_penalty = sum((p ** 2).sum() for p in model.parameters())
        loss += l1_lambda * l1_penalty + l2_lambda * l2_penalty

        # Backward pass and optimize
        loss.backward()
        optimizer.step()

        # Print statistics
        running_loss += loss.item()
        predicted = (torch.sigmoid(outputs) > 0.5).float()
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    train_losses_finetune.append(running_loss / len(train_loader))
    train_accuracies_finetune.append(correct / total)

    val_loss, val_accuracy, _, _, _, _ = evaluate(model, val_loader, criterion,
device)
    val_losses_finetune.append(val_loss)
    val_accuracies_finetune.append(val_accuracy)
    print(f'Epoch {epoch+1}, Train Loss: {train_losses_finetune[-1]}, Train
Accuracy: {train_accuracies_finetune[-1]}, Val Loss: {val_losses_finetune[-
1]}, Val Accuracy: {val_accuracies_finetune[-1]}')

# Save the model
torch.save(model.state_dict(), 'modified_model_resnet50_finetuned.pth')
```


- Appendix VI VGG16

```
# Load the pre-trained VGG16 model
base_model = models.vgg16(weights=models.VGG16_Weights.IMAGENET1K_V1)

# Freeze the base model layers
for param in base_model.parameters():
    param.requires_grad = False

# Modify the VGG16 model
class ModifiedVGG16(nn.Module):
    def __init__(self, base_model):
        super(ModifiedVGG16, self).__init__()
        self.base_model = base_model.features
        self.custom_layers = nn.Sequential(
            nn.Conv2d(512, 256, kernel_size=3, padding=1),
            nn.BatchNorm2d(256),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Dropout(0.5),
            nn.Conv2d(256, 128, kernel_size=3, padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Dropout(0.5),
            nn.AdaptiveAvgPool2d((1, 1)),  # Adaptive average pooling to handle variable sizes
            nn.Flatten()
```

```
        )
        self.fc = nn.Sequential(
            nn.Linear(128, 1024),  # Adjusted to match the flattened size
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(1024, 1)
        )

    def forward(self, x):
        x = self.base_model(x)
        x = self.custom_layers(x)
        x = self.fc(x)
        return x

# Create the modified model
model = ModifiedVGG16(base_model)

# Define loss function and optimizer
criterion = nn.BCEWithLogitsLoss()
optimizer = optim.Adam(model.parameters(), lr=0.0001)

# L1 and L2 regularization parameters
l1_lambda = 1e-5
l2_lambda = 1e-4

# Move model to GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Lists to store metrics for initial training phase
train_losses_initial = []
val_losses_initial = []
train_accuracies_initial = []
val_accuracies_initial = []

# Lists to store metrics for fine-tuning phase
train_losses_finetune = []
val_losses_finetune = []
train_accuracies_finetune = []
val_accuracies_finetune = []

def evaluate(model, data_loader, criterion, device):
```

```
    model.eval()
    running_loss = 0.0
    correct = 0
    total = 0
    all_labels = []
    all_predictions = []
    with torch.no_grad():
        for images, labels in data_loader:
            images = images.to(device)
            labels = labels.to(device).float().view(-1, 1)
            outputs = model(images)
            loss = criterion(outputs, labels)
            # L1 and L2 penalties
            l1_penalty = sum(p.abs().sum() for p in model.parameters())
            l2_penalty = sum((p ** 2).sum() for p in model.parameters())
            loss += l1_lambda * l1_penalty + l2_lambda * l2_penalty
            running_loss += loss.item()
            predicted = (torch.sigmoid(outputs) > 0.5).float()
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
            all_labels.extend(labels.cpu().numpy())
            all_predictions.extend(predicted.cpu().numpy())
    cm = confusion_matrix(all_labels, all_predictions)
    f1 = f1_score(all_labels, all_predictions)
    precision = precision_score(all_labels, all_predictions)
    recall = recall_score(all_labels, all_predictions)
    return running_loss / len(data_loader), correct / total, cm, f1, precision,
recall

# Training loop with all layers frozen
num_epochs = 10
for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0

    for images, labels in train_loader:
        images = images.to(device)
        labels = labels.to(device).float().view(-1, 1)

        # Zero the parameter gradients
```

```
        optimizer.zero_grad()

        # Forward pass
        outputs = model(images)
        loss = criterion(outputs, labels)
        # L1 and L2 penalties
        l1_penalty = sum(p.abs().sum() for p in model.parameters())
        l2_penalty = sum((p ** 2).sum() for p in model.parameters())
        loss += l1_lambda * l1_penalty + l2_lambda * l2_penalty

        # Backward pass and optimize
        loss.backward()
        optimizer.step()

        # Print statistics
        running_loss += loss.item()
        predicted = (torch.sigmoid(outputs) > 0.5).float()
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    train_losses_initial.append(running_loss / len(train_loader))
    train_accuracies_initial.append(correct / total)

    val_loss, val_accuracy, _, _, _, _ = evaluate(model, val_loader, criterion,
device)
    val_losses_initial.append(val_loss)
    val_accuracies_initial.append(val_accuracy)

    print(f'Epoch {epoch+1}, Train Loss: {train_losses_initial[-1]}, Train
Accuracy: {train_accuracies_initial[-1]}, Val Loss: {val_losses_initial[-1]},
Val Accuracy: {val_accuracies_initial[-1]}')

# Unfreeze the last 10 layers
for param in list(model.parameters())[-10:]:
    param.requires_grad = True

# Fine-tuning loop
num_finetune_epochs = 5
for epoch in range(num_finetune_epochs):
    model.train()
    running_loss = 0.0
    correct = 0
```

```
    total = 0

    for images, labels in train_loader:
        images = images.to(device)
        labels = labels.to(device).float().view(-1, 1)

        # Zero the parameter gradients
        optimizer.zero_grad()

        # Forward pass
        outputs = model(images)
        loss = criterion(outputs, labels)
        # L1 and L2 penalties
        l1_penalty = sum(p.abs().sum() for p in model.parameters())
        l2_penalty = sum((p ** 2).sum() for p in model.parameters())
        loss += l1_lambda * l1_penalty + l2_lambda * l2_penalty

        # Backward pass and optimize
        loss.backward()
        optimizer.step()

        # Print statistics
        running_loss += loss.item()
        predicted = (torch.sigmoid(outputs) > 0.5).float()
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    train_losses_finetune.append(running_loss / len(train_loader))
    train_accuracies_finetune.append(correct / total)

    val_loss, val_accuracy, _, _, _, _ = evaluate(model, val_loader, criterion,
device)
    val_losses_finetune.append(val_loss)
    val_accuracies_finetune.append(val_accuracy)
    print(f'Epoch {epoch+1}, Train Loss: {train_losses_finetune[-1]}, Train
Accuracy: {train_accuracies_finetune[-1]}, Val Loss: {val_losses_finetune[-
1]}, Val Accuracy: {val_accuracies_finetune[-1]}')

# Save the model
torch.save(model.state_dict(), 'modified_model_vgg16_finetuned.pth')
```


- Appendix VII Optimized CNN

```
def build_model(hp):
    model = Sequential()

    # Regularization parameter
    l2_reg = 0.001  # L2 regularization parameter, can be tuned

    # Tune the number of convolutional layers
    num_conv_layers = hp.Int('num_conv_layers', min_value=1, max_value=5, step=1)

    # Add the input layer with L2 regularization
    model.add(layers.Conv2D(filters=hp.Int('filters_0', min_value=32, max_value=128, step=32),
                            kernel_size=hp.Choice('kernel_size_0', values=[3, 5]),
                            activation='relu',
                            kernel_regularizer=regularizers.l2(l2_reg),
                            input_shape=(250, 250, 3)))  # Assuming input images are 150x150 with 3 channels (RGB)
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    model.add(layers.Dropout(rate=hp.Choice('dropout_0', values=[0.2, 0.3, 0.4])))  # Add dropout layer after pooling

    # Add intermediate convolutional layers with L2 regularization and dropout
    for i in range(num_conv_layers):
        model.add(layers.Conv2D(filters=hp.Int(f'filters_{i+1}', min_value=32, max_value=128, step=32),
                                kernel_size=hp.Choice(f'kernel_size_{i+1}', values=[3, 5]),
                                activation='relu',
                                kernel_regularizer=regularizers.l2(l2_reg)))
        model.add(layers.MaxPooling2D(pool_size=(2, 2)))
        model.add(layers.Dropout(rate=hp.Choice(f'dropout_{i+1}', values=[0.2, 0.3, 0.4])))  # Add dropout layer

    model.add(layers.Flatten())
```

```
    # Add a fully connected layer with L2 regularization and dropout
    model.add(layers.Dense(units=hp.Int('units_dense',          min_value=128,
max_value=512, step=128),
                           activation='relu',
                           kernel_regularizer=regularizers.l2(l2_reg)))
    model.add(layers.Dropout(rate=hp.Choice('dropout_dense',    values=[0.2,
0.3, 0.4])))  # Add dropout layer

    # Add the output layer
    model.add(layers.Dense(units=1, activation='sigmoid'))

    # Tune the learning rate
    hp_learning_rate = hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-4])

    # Compile the model

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=hp_learni
ng_rate),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])

    return model


# Define the search space and tuner (unchanged)
tuner = RandomSearch(build_model,
                     objective='val_accuracy',
                     max_trials=18,
                     directory='CNN_training',
                     project_name='training_cnn')

# Start the search for the best model
tuner.search(train_loader, epochs=10, validation_data=val_loader)

best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]
print(best_hps.values)

model_CNN = tuner.hypermodel.build(best_hps)

# Define the EarlyStopping callback (unchanged)
early_stopping = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)
```

```
# Retrain the model
history = model_CNN.fit(train_loader, epochs=15, validation_data=val_loader, callbacks=[early_stopping])

# Save the model and the training history
model_CNN.save("model_CNN_optimized_2.h5")
```