MINISTRY OF EDUCATION

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

FACULTY OF AUTOMATION AND COMPUTER SCIENCE
COMPUTER SCIENCE DEPARTMENT

PNEUMONIA DETECTION USING FEDERATED LEARNING

LICENSE THESIS

Graduate: **Kruk Zsolt- Istvan**

Supervisor: **SL dr. ing. Dan Gota**

2023

MINISTRY OF EDUCATION

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

FACULTY OF AUTOMATION AND COMPUTER SCIENCE
COMPUTER SCIENCE DEPARTMENT

DEAN,
Prof. dr. eng. Liviu MICLEA

HEAD OF DEPARTMENT,
Prof. dr. eng. Honoriu VALEAN

Graduate: **Kruk Zsolt-Istvan**

PNEUMONIA DETECTION USING FEDERATED LEARNING

1. **Project proposal:** *Pneumonia detection using federated learning is a decentralized approach that uses local data from multiple devices to train a machine learning model without the need for data centralization. The dataset consists of two sets of Chest X-Ray images. One set has signs of pneumonia and the other did not.*

2. **Project contents:** *Presentation page, advisor's evaluation, Introduction, Project Objectives, Bibliographical Research, Analysis and Theoretical Foundation, Detailed Design and Implementation, Testing and Validation, User's Manual, Conclusions, Bibliography, Appendices.*

3. **Place of documentation**: Technical University of Cluj-Napoca, Automation Department

4. **Consultants**: dr. eng. Stefan Ioana, Petrescu Bianca-Ionela, eng. Mocanu Vlad Alexandru, eng. Grigoras Cosmin Dragos

5. **Date of issue of the proposal:** December 8, 2022

6. **Date of delivery:** July 6, 2023

Graduate: Kruk Zsolt

Supervisor:

MINISTRY OF EDUCATION

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

FACULTY OF AUTOMATION AND COMPUTER SCIENCE
COMPUTER SCIENCE DEPARTMENT

Declarație pe propria răspundere privind autenticitatea lucrării de licență

Subsemnatul(a) KRUK ZSOLT ISTVAN, legitimat(ă) cu CI seria XM nr. 153350 CNP 5001111245034, autorul lucrării elaborată în vederea susținerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ ENGLEZĂ din cadrul Universității Tehnice din Cluj-Napoca, sesiunea IULIE a anului universitar 2022-2023, declar pe propria răspundere că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării, și în bibliografie.

Declar că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență.

In cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență*.

Data	Nume, Prenume
1.07.2023	KRUK ZSOLT-ISTVAN

Semnătura

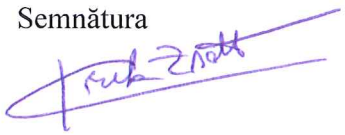


Table of contents

Chapter 1. Introduction 6
1.1. Background and Motivation 6
1.1.1. Pneumonia Diagnosis 6
1.1.2. Deep Learning and Medical Imaging 6
1.1.3. Federated Learning 6
1.1.4. Research Gap 6
1.2. Problem Statement 7

Chapter 2. Project Objectives 8
2.1. Objectives 8
2.2. Research Questions 8
2.3. Scope and Limitations 8

Chapter 3. Bibliographic Research 9
3.1. Overview of Pneumonia and its Detection 9
3.2. Machine Learning Applications in Healthcare 10
3.3. Healthcare Applications of Federated Learning 11
3.4. Research Studies and their Conclusions 12

Chapter 4. Analyisis and Theoretical Foundation 14
4.1. Research Design 14
4.2. Data Collection and Preprocessing 16
4.3. Convolutional Networks and Machine Learning 17
4.4. Model Architecture and Selection 21
4.4.1. VGG16 22
4.4.2. ResNet50 22
4.4.3. InceptionV3 23
4.4.4. Model Selection 24
4.5. Federated Learning Approach 25
4.5.1. Overview of Federated Learning 25
4.5.2. Communication between Clients and Server 26
4.5.3. Model Aggregation 29
4.5.4. Evaluation Metrics 31

Chapter 5. Detailed Design and Implementation 34
5.1. Python and the Libraries Used 34

5.2. Overview of the Dataset 35
5.3. Image Preprocessing 38
5.4. Transfer Learning 40
5.5. Federated Learning Implementation 45
5.6. Web-based Pneumonia Detection Using Flask and Swagger 51
5.7. Containerizing the Pneumonia Detection System with Docker 54
5.8. Discussion of the Results 56

Chapter 6. Testing and Validation 59
6.1. Performance Metrics of the Models 59
6.2. Transfer Learning with ResNet50 61
6.3. Web Application Testing 64

Chapter 7. User's Manual 69

Chapter 8. Conclusions 72
8.1. Summary of the Findings 72
8.2. Contributions of the Study 72
8.3. Limitations and Future Work 73

Bibliography 75

Chapter 1. Introduction

1.1. Background and Motivation

1.1.1. Pneumonia Diagnosis

Pneumonia is a respiratory infection that affects the lungs producing swelling and fluid buildup in the air sacs. It is the leading cause of mortality and morbidity worldwide, causing an estimated 2 million deaths each year, particularly in young children, elderly adults, and immunocompromised individuals [1]. It is crucial to get an early and correct diagnosis of pneumonia to treat it effectively and avoid complications. Medical images, especially chest X-rays, are commonly used diagnostic tools for pneumonia, with characteristic features such as consolidation, airspace opacities, and pleural effusion [2]. Accurate interpretation of these images can be challenging and time-consuming especially in resource-constrained settings.

1.1.2. Deep Learning and Medical Imaging

Recent advances in machine learning, and specifically deep learning, has the potential to significantly increase the accuracy and speed of pneumonia identification from medical imaging [2][3]. Deep learning models, like convolutional neural networks (CNNs), are capable of automatically learning information from medical images and making precise predictions. However, centralized machine learning models often require access to large amounts of data, which may be difficult to obtain, especially for rare diseases or in regions with limited resources. Additionally, privacy and security can become an issue when sensitive medical information is shared.

1.1.3. Federated Learning

Federated learning is an innovative approach to machine learning that addresses these challenges by allowing machine learning models to be trained on distributed data sources without the need for centralized storage or sharing of data [4]. In federated learning, multiple data sources collaborate to train a shared model while preserving individual control over their own data. Federated learning has proven the efficacy in a variety of applications, including image classification, natural language processing, autonomous vehicles, and advertising [5]. In the context of pneumonia detection, federated learning has the potential to improve diagnostic accuracy and speed while maintaining patient data privacy and security, as data remains on the local device or server and is never shared.

1.1.4. Research Gap

Despite the potential benefits of federated learning for medical imaging, there are currently limited studies on the use of this approach for pneumonia detection. The purpose of this bachelor's thesis is to investigate the feasibility and performance of pneumonia detection using federated learning. Specifically, whether federated learning can improve the accuracy and the efficiency of pneumonia detection models compared to traditional centralized machine learning models, and which factors such as model architecture and hyperparameters influence the performance of federated learning-based pneumonia detection models. The results of this

study may have significant implications for the development and implementation of improved diagnostic tools for pneumonia, ultimately contributing to improved healthcare outcomes for patients.

1.2. Problem Statement

Pneumonia remains a major health problem worldwide, with millions of cases and hundreds of thousands of deaths reported each year. Accurate and timely diagnosis of pneumonia is essential for effective treatment and management of the disease. Interpreting chest X-rays can be difficult, especially in patients with mild infections or pre-existing lung disease.

While the recent advances in deep learning have shown promise in improving accuracy and speed of pneumonia detection, these models often require access to large amounts of data. This data can be difficult to obtain, particularly in regions with limited resources. Additionally, sharing this highly sensitive medical data can raise concerns about the privacy and security of the patient.

The problem addressed by this thesis is to investigate the feasibility and performance of pneumonia detection using federated learning. The following questions are addressed by this study.

- How does federated learning perform compared to typical centralized machine learning models for pneumonia detection?
- What are the potential benefits and boundaries of using federated learning for pneumonia detection?
- How can federated learning be optimized to improve the accuracy and efficiency of pneumonia detection while maintaining patient privacy and security?

By answering these research questions, this study aims to contribute to the development of more effective and secure methods for pneumonia diagnosis using deep learning and federated learning.

Chapter 2. Project Objectives

2.1. Objectives

The main objectives of this project are:

- To develop a pneumonia detection model using federated learning that is accurate and efficient in processing medical images.
- To evaluate the performance of the model on distributed datasets to determine the effectiveness in detecting pneumonia cases.
- To compare the performance of different transfer learning models and chose the best one for pneumonia detection using federated learning.
- To investigate the feasibility and efficiency of federated learning for medical image processing and to pinpoint any problems and potential remedies.
- To demonstrate the potential of federated learning to improve privacy and security in medical data analysis by keeping patient information decentralized and secure.
- To use a federated learning framework that enables the model to be trained on multiple devices and data sources without centralized data storage or access.
- To develop a user-friendly interface for healthcare professionals to interact with the model and verify its results.

2.2. Research Questions

The research questions, in the context of this thesis, seek to investigate the feasibility and efficacy of using federated learning for pneumonia diagnosis, identifying potential difficulties and solutions, as well as comparing the performance of the federated learning model with conventional centralized model. The thesis will cover a variety of research questions, including:

- How accurate is the federated learning-based pneumonia detection model?
- What are the advantages and disadvantages of using federated learning for medical image analysis?
- How can the privacy and security of patient data be ensured in a federated learning system for pneumonia detection?
- What are the technical requirements needed for implementing a federated learning framework for medical image analysis?
- How can the pneumonia detection model using federated learning be integrated into clinical settings to improve patient outcomes and healthcare delivery?

2.3. Scope and Limitations

The scope of this thesis focuses on developing a pneumonia detection model using federated learning and evaluating its performance on a distributed dataset. Specifically, the thesis will investigate the feasibility and efficacy of using federated learning for medical image analysis and compare the performance of the model with conventional centralized machine learning approaches. The thesis will also explore potential privacy-preserving techniques such model aggregation which ensures the confidentiality of patient data. The research will be limited to the use of chest X-rays as the primary medical images for pneumonia detection, and the use of Flower as the main library for federated learning. Lastly, the research may be limited by the availability and quality of data from different sources, and the computational resources required to train the model using federated learning.

Chapter 3. Bibliographic Research

3.1. Overview of Pneumonia and its Detection

Pneumonia is a lung infection that affects roughly 450 million to 500 million people each year, causing over 4 million deaths annually. It is caused by different agents such as bacteria, viruses, and fungi. The annual incidence of pneumonia in developed countries is estimated to be 33 per 10000 children under the age of 5 and 14.5 per 10000 children between the ages of 5 and 16 [6]. The World Health Organization (WHO) reports that pneumonia is the primary cause of death in children under the age of five, killing an estimated 1.2 million of them each year. 18% of all fatalities of children under the age of five occur due to this worldwide [6]. Pneumonia symptoms might differ depending on the cause of the infection, the person's age, and general condition. Some common symptoms of pneumonia include persistent coughing that may produce phlegm, high fever, shortness of breath, chest pain, fatigue, sweating and confusion [7]. For efficient treatment and the prevention of complications, early diagnosis of pneumonia is crucial. There are multiple ways of detecting pneumonia, computed tomography (CT), ultrasound imaging, blood test, sputum test, bronchoscopies, but the main method in diagnosing pneumonia remains chest radiography (Chest X-Ray) (Figure 3.1). However, it does have certain restrictions, like inter-observer variability and radiation exposure. Therefore, there is a need for more precise and effective diagnostic procedures for the identification of pneumonia.

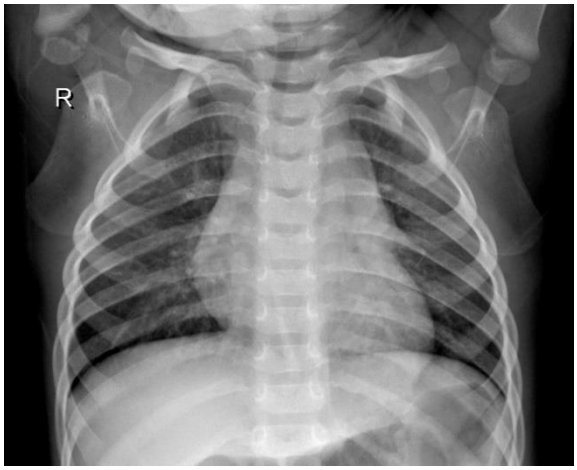


Figure 3.1 Chest X-Ray

In recent years, machine learning has emerged to be a valuable tool used for pneumonia detection. Deep learning-based approaches have shown promising results in automated pneumonia detection from Chest X-Rays. These methods use convolutional neural networks (CNNs) to identify the characteristics that set pneumonia apart from healthy lungs. Highly skilled radiologists can detect pneumonia; however, the process can be time-consuming and frequently results in possible conflict between radiologists. Considering this, deep learning offers a quick, highly precise method of objectively diagnosing pneumonia using medical imaging. [8].

3.2. Machine Learning Applications in Healthcare

In the healthcare industry, machine learning has developed into a powerful tool that is altering a number of aspects of the industry. This chapter explores the various applications of machine learning in healthcare, highlighting its potential to improve operational efficiency as well as patient care, diagnosis, and treatment.

In the study of medical imaging, machine learning algorithms have achieved extraordinary results. X-rays, CT scans, MRIs, and images from histopathology are just a few of the imaging modalities they can accurately identify and classify abnormalities in. Radiologists can use these algorithms to detect diseases including cancer, cardiovascular disease, and neurological issues early on. According to studies by Esteva et al. (2017) [9] and Gulshan et al. (2016) [10], machine learning is successful in classifying skin cancer and identifying diabetic retinopathy.

Machine learning enables predictive analytics, which forecasts the onset of illnesses, the effectiveness of treatments, and the identification of high-risk patients using patient data. Algorithms can examine genetic data, physiological signals, and electronic health records to build personalized risk models for diseases including diabetes, heart disease, and sepsis. For instance, Beaulieu-Jones et al. (2018) [11] were able to predict patient deterioration using electronic health data and machine learning.

Drug development and discovery are greatly accelerated by machine learning. Huge volumes of biological data, including genomes, proteomics, and chemical structures, can be analyzed by algorithms to pinpoint prospective drug targets and create fresh therapies. Machine learning is used in precision medicine to find biomarkers, forecast treatment outcomes, and regulate drug dose. Precision medicine customizes treatments based on unique patient features. The use of machine learning in drug development and precision oncology are highlighted in notable works by Clémence Réda et al. (2020) [12] and Robert F. Mutphy et al. (2020) [13], accordingly.

Healthcare operations are improved by machine learning algorithms through better resource allocation, more effective workflow, and cost savings. To effectively distribute staff and resources, hospitals can foresee patient demand using predictive models. Machine learning also assists in the detection of fraud, the management of the revenue cycle, and the streamlining of administrative procedures. Machine learning is used in a Stephen Bacchi et al. (2018) [14] study to predict patient length of stay and optimize bed allocation.

Despite its many advantages, machine learning in healthcare also poses problems and ethical dilemmas. Important areas of concern include preserving algorithmic fairness, protecting data privacy, and deciphering black-box algorithms. To protect patient safety and avoid inequities in healthcare delivery, it is important to pay close attention to the interpretability of machine learning algorithms and their potential for bias. The difficulties of implementing machine learning models in clinical practice and the significance of addressing ethical issues are covered in research by Irene Y. Chen et al. (2018) [15].

3.3. Healthcare Applications of Federated Learning

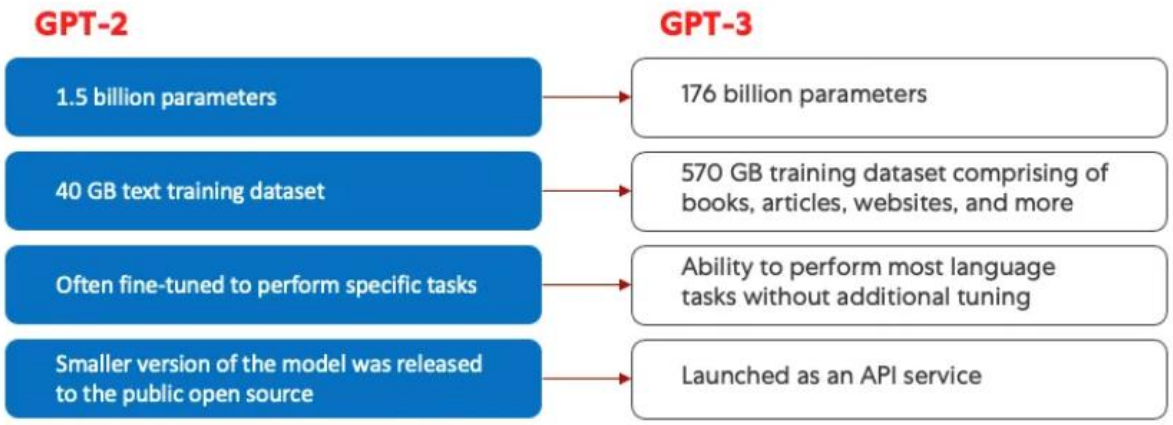


Figure 3.2 ChatGPT2 vs ChatGPT3.

The significance of data science in industrial engineering and medical imaging is made clear by advancements in storage capacity and processing power. Recent years have seen an increasing demand and use of artificial intelligence, machine learning, smart production, and deep learning. Nevertheless, there are two significant obstacles to the development of data science. To start, the most significant aspect is data governance. Some information is privatized due to legal reasons. Users are now the sole owners of their own data because of the General Data Protection Regulation (GDPR) (EU, 2018) being implemented [16]. Another main challenge are the data silos. These put a limit on the development of data science since more training data would improve training performance. For example, compared to the earliest ChatGPT 2, which used 1.5 billion parameters and 40 GB of text training dataset, was only able to perform some fine-tuned specific tasks, ChatGPT 3 uses 176 billion parameters and 570 GB of trainable dataset and can perform most language tasks (Figure 3.2) [17]. Thus, the emergence of federated learning can overcome these challenges in both industrial engineering and medical imaging.

Federated learning is a decentralized machine learning strategy that enables multiple entities to cooperatively train shared data without disclosing their respective data to one another. The advantages it has over traditional centralized machine learning are preserving data privacy, reducing communication costs, and enabling the training of models using distributed data sources, thus removing the need for data silos. It uses multiple clients (e.g. mobile devices, institutions, organizations, etc.) to communicate with one or more central servers for aggregating the trained models. While Google originally proposed federated learning in 2016 to predict user text input on thousands of Android devices while retaining the data on the devices, it recently gained traction for healthcare applications [18]. By using a consensus model, federated learning makes it possible to gather insights collectively without transferring patient data outside of the institutions where they are housed. As opposed to this, the ML process takes place locally at each participating institution, and only model characteristics (such as parameters and gradients) are transferred as depicted in Figure 3.3.

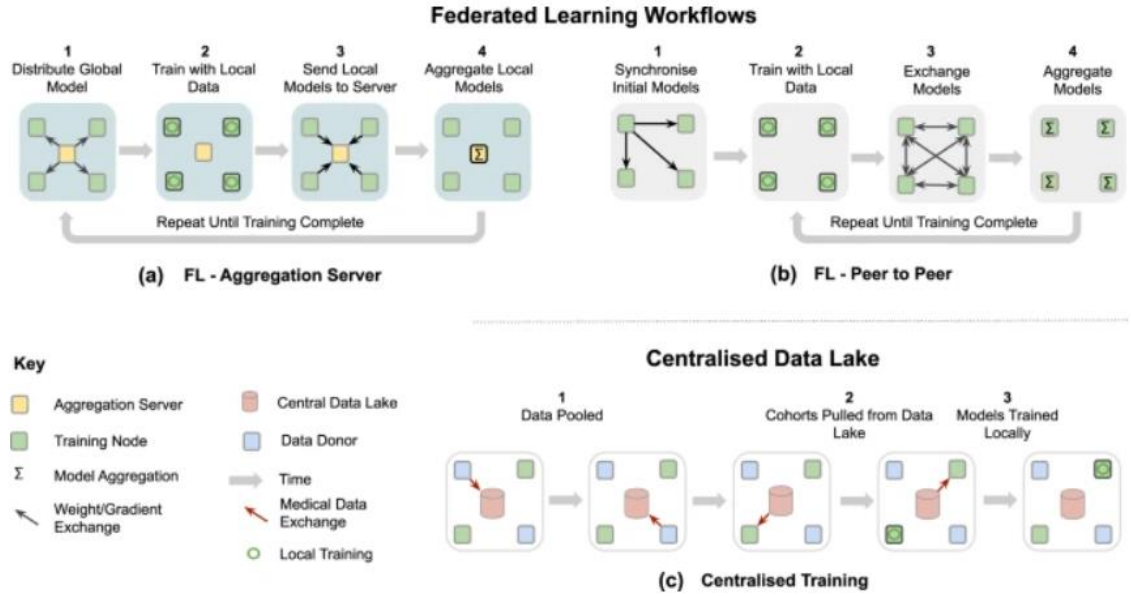


Figure 3.3 Example FL workflows and difference from learning on a centralized data lake.

The distinctions between centralized machine learning (c), federated learning with peer-to-peer communication (b), and federated learning with an aggregation server (a) are shown in Figure 3.3. The most popular federated learning workflow uses an aggregation server, in which a network of training nodes receives the global model, resubmits their partially trained models to a centralized server for aggregation, and then continues training on the consensus model that the server returns. Another, slightly less popular, method for federated learning is peer-to-peer. In this, each training node exchanges its partially trained models with some or all of its peers and each does its own aggregation. Peer-to-peer learning is another, slightly less popular, method of federated learning. In this, each training node performs its own aggregation and exchanges its partially trained model with some or all of its peers. Finally, centralized training refers to the standard non-federated learning workflow in which data-acquiring sites contribute their data to a central Data Lake from which they and others can extract data for local, independent training [19].

While federated learning has the potential to overcome the limitations of approaches that require a single pool of centralized data, it still requires rigorous technical analysis to make sure the algorithm is working smoothly without endangering the patient privacy or safety. Thus, a successful implementation of federated learning could have a significant impact on the ability to practice precision medicine on a large scale, resulting in models that produce unbiased decisions, and are sensitive to rare diseases while respecting governance and privacy concerns.

3.4. Research Studies and their Conclusions

Numerous studies have investigated the application of federated learning for the diagnosis of pneumonia using medical images. For example, Khan et al. [20] suggested a federated learning strategy using transfer learning for exploiting chest X-rays to detect pneumonia. They compared different pre-trained models including VGG16, ResNet50, AlexNet and custom CNN, and achieved high accuracy while preserving data privacy. Similar,

Theodora S. Brisimi et al. [21] developed a federated learning framework that allowed various institutions, each with their own private data, to cooperate and converge to a common predictive model, used to predict hospitalizations, mortality, and length of ICU stay due to cardiac events, without sharing raw data. They created an iterative cluster Prima Dual Splitting (cPDS) algorithm to solve large-scale sparse Support Vector Machine (sSVM) problems in a decentralized manner, and they got results that were comparable to those of a centralized approach while achieving faster convergence.

Aaisha Makkar et al. [22] explored various aggregation techniques for integrating the locally trained models in another study. They put forth a new aggregation technique they called secureFed, which is based on the Markov method. Andrey Markov created stochastic models called Markov chains that demonstrate the likelihood of a sequence of events taking place given the outcome of the previous event. Once the current state of the vector is added to the Markov chain, regardless of whether the patient has COVID, the prediction is stored in a temporary matrix that the server uses during aggregation. Two matrices are kept while the chain is being updated by the clients: one is made up of the initial vectors generated during local training, and the other is a temporary matrix created during the Markov chain's instant prediction.

In a different study, Chaoyang He et al. [23] suggested utilizing AutoFL and NAS to automate the model design for each local workstation inside a distributed federated learning network. FedNAS (automatic federated learning with Neural Architecture Search) greatly outperformed a more general averaging technique using a predefined DenseNet model in terms of accuracy, run time, and parameter size.

In conclusion, federated learning-based pneumonia detection is a promising area of study that has the potential to revolutionize the field of medical imaging. Federated learning enables the development of precise and reliable pneumonia detection models while protecting data privacy and security by utilizing the capabilities of deep learning and distributed computing.

The efficiency of federated learning for pneumonia identification utilizing medical pictures, such as chest X-rays and CT scans, has been shown in numerous research. These research have demonstrated that while resolving the issues of data heterogeneity and privacy preservation, federated learning can achieve equivalent or even greater performance compared to classic centralized learning systems.

Federated learning can be a useful tool for pneumonia diagnosis in real-world healthcare settings, but there are still several obstacles and constraints that need to be overcome. These include the models' robustness, scalability, and generalizability as well as any potential moral and legal issues pertaining to data ownership and patient privacy. Further investigation is required to create more sophisticated and scalable federated learning frameworks and algorithms, as well as to investigate the potential uses of federated learning in healthcare settings other than pneumonia diagnosis. In order to ensure the ethical and responsible use of federated learning in healthcare and to address the possible risks and challenges related with this technology, coordination between researchers, doctors, and policymakers is also essential.

Overall, federated learning holds great potential for the detection of pneumonia and other medical applications, and in the years to come, it is expected to play a bigger part in medical imaging and diagnostics.

Chapter 4. Analysis and Theoretical Foundation

4.1. Research Design

This thorough and innovative study's aim is to delve into the fascinating world of healthcare applications and shed light on the promised efficacy of using federated learning as a potentially game-changing technique for pneumonia identification. By investigating this novel strategy, we hope to determine if federated learning can indeed improve pneumonia detection efficiency and accuracy while respecting the fundamental values of data privacy and security.

Understanding the overall significance of pneumonia detection within healthcare is crucial. An extremely serious respiratory infection of the lungs known as pneumonia causes a variety of problems, including death. Pneumonia can be accurately and quickly detected, which is essential for enabling immediate therapeutic interventions, enhancing patient outcomes, and ultimately saving lives.

Federated learning has become a ground-breaking paradigm in recent years, showing considerable potential in a variety of fields, including healthcare. With the help of this innovative method, machine learning models may be jointly trained on decentralized data sources while maintaining the confidentiality of private data. Healthcare organizations may be able to take advantage of the enormous amounts of dispersed data that are available across several organizations by utilizing the power of federated learning, building a strong and cooperative framework for pneumonia detection.

Whether federated learning may act as a catalyst for improving both the accuracy and efficiency of pneumonia detection is the main research topic that underlies this study. Additionally, the benefit of decentralized training provided by federated learning eliminates the requirement for data centralization. As data is kept securely maintained within each institution's local environment, this decentralized method reduces privacy concerns related to exchanging sensitive patient information. Federated learning upholds patient record confidentiality and complies with legal and ethical requirements while facilitating collaborative research and knowledge sharing by placing a high priority on data privacy and security.

This study will examine potential difficulties and restrictions related to federated learning's application in pneumonia detection in addition to assessing its effectiveness. These might involve problems with model aggregation methods, communication overhead, and data heterogeneity. We hope to offer useful insights into the practical ramifications of adopting federated learning in actual healthcare settings by identifying and addressing these roadblocks.

In the end, our research hopes to expand our understanding of pneumonia diagnosis and open the door for revolutionary improvements in healthcare. We aim to build a future where accurate, efficient, and privacy-preserving pneumonia diagnosis becomes a reality, benefiting healthcare professionals, researchers, and, most importantly, patients globally by exploring the potential of federated learning.

Based on the literature review, we hypothesize the following:

- While maintaining data security and privacy, the proposed federated learning approach may identify pneumonia with an accuracy that is on par with or better than that of conventional centralized learning systems.
- By expanding the pool of participating clients and improving the communication protocols, the suggested federated learning approach's performance can be enhanced.

The scope of this study is limited to the use of deep learning models and federated learning approaches for pneumonia detection. We will use publicly available chest X-ray images from Kaggle provided by the University of California San Diego (UCSD) to train and test our models. We will also limit the study to binary classification of pneumonia vs. normal cases, although the proposed approach can be extended to multi-class classification and other healthcare applications.

The variability and poor quality of the data sources, the federated learning approach's communication costs and bandwidth restrictions, and the meager computer resources available for model training and testing are among the study's drawbacks.

To accomplish its goals, this study will combine experimental and analytical techniques. Phases of the approach include the following:

- **Data collection and preprocessing:** Chest X-ray pictures will be gathered from the UCSD databases and processed before being used to train and test the algorithms. Techniques like image scaling, normalization, and augmentation will be covered in this.

- **Model architecture and selection:** We will investigate various deep learning pre-trained architectures and models such as VGG16, Inception, ResNet and modify them for the diagnosis of pneumonia. Additionally, we'll decide which model is the best depending on how well it performs and how sophisticated it is.

- **Federated learning approach:** To train pneumonia detection models on distributed data sources while maintaining data security and privacy, we will create a federated learning strategy. We will use Flower Federated Learning (FL) to create the clients, server, and communication protocols.

- **Evaluation metrics:** We will evaluate the effectiveness of the pneumonia detection algorithms using metrics including accuracy, precision, recall, and F1 score. To guarantee the robustness and generalizability of the models, we will also employ cross-validation approaches.

Ethics-related principles and considerations, such as data privacy, informed consent, and transparency, will be followed in this study.

4.2. Data Collection and Preprocessing

For the purpose of developing reliable and accurate pneumonia detection algorithms, high-quality and diversified data must be readily available. We will discuss the methods used to get the data and to preprocess it in this chapter so that it is reliable and appropriate for our investigation.

The Guangzhou Women and Children's Medical Center (GWCMC) provided the dataset, which was later acquired from Paul Mooney via Kaggle. This database provides access to many publicly available chest X-ray images that have been expertly annotated by radiologists. The GWCMC database focuses specifically on pediatric chest X-ray images and features normal cases, bacterial and viral pneumonia cases.

Before training our pneumonia detection models, we need to preprocess the data to ensure its quality, consistency, and compatibility. The gathered chest X-ray pictures will go through the following preprocessing methods:

- **Image Resizing**: To ensure a consistent input size for our models, we will resize all the chest X-ray images to a predefined resolution. This resizing step will assist standardize the image size and ensure that the models can handle them consistently.

- **Image Normalization**: Image normalization will be used to increase the convergence and stability of our models during training. The pixel values in the images are adjusted to have an identical range or distribution using this technique. Normalization can help to reduce the impact of differences in lighting and contrast across photos.

- **Data Augmentation:** Data augmentation is an essential approach for increasing the size of the training dataset and introducing variation. By adding changes to the X-ray images of the chest such as rotation, scaling, flipping, and cropping, we may increase the variety of the training data and improve the model's capacity to generalize to previously unseen examples.

- **Balancing the Dataset:** Imbalanced datasets, in which the number of pneumonia patients differs greatly from the number of normal cases, can lead to biased models. As a result, we will ensure that the final dataset is balanced by randomly oversampling the minority class and under sampling the majority class. This method prevents the models from favoring one class over the other during training.

It is critical to ensure the quality and reliability of the collected data when developing reliable pneumonia detection models. Expert radiologists identified the chest X-ray pictures in the GWCMC database, giving credible ground truth annotations. This knowledge ensures that the labels correctly indicate the presence or absence of pneumonia in the photos. We will carefully evaluate the acquired dataset to detect and eliminate any photographs of low quality, artifacts, or cases that are incorrectly labeled. This procedure will help to maintain the dataset's integrity and reliability, ensuring that our models are trained on high-quality data.

We will divide the acquired dataset into training, validation, and testing sets to evaluate the performance of our pneumonia detection models. The training set will be utilized to train the models, the validation set for hyperparameter tuning and model selection, and the testing set for evaluating the final performance of the chosen models.

4.3. Convolutional Networks and Machine Learning

CNNs have proven to be more effective than conventional approaches at detecting pneumonia. They are exceptional in detecting subtle abnormalities and detailed patterns in medical photos, enabling precise and trustworthy diagnosis. In order to diagnose pneumonia, CNNs' capacity to learn hierarchical representations from unprocessed pixel data is very useful because it allows the network to automatically extract pertinent features without the need for manual feature engineering.

Convolutional networks are made up of several layers, each of which serves a particular function during the learning process. The fundamental CNN building pieces that were utilized to create the models for pneumonia detection are:

- **Convolutional Layer**

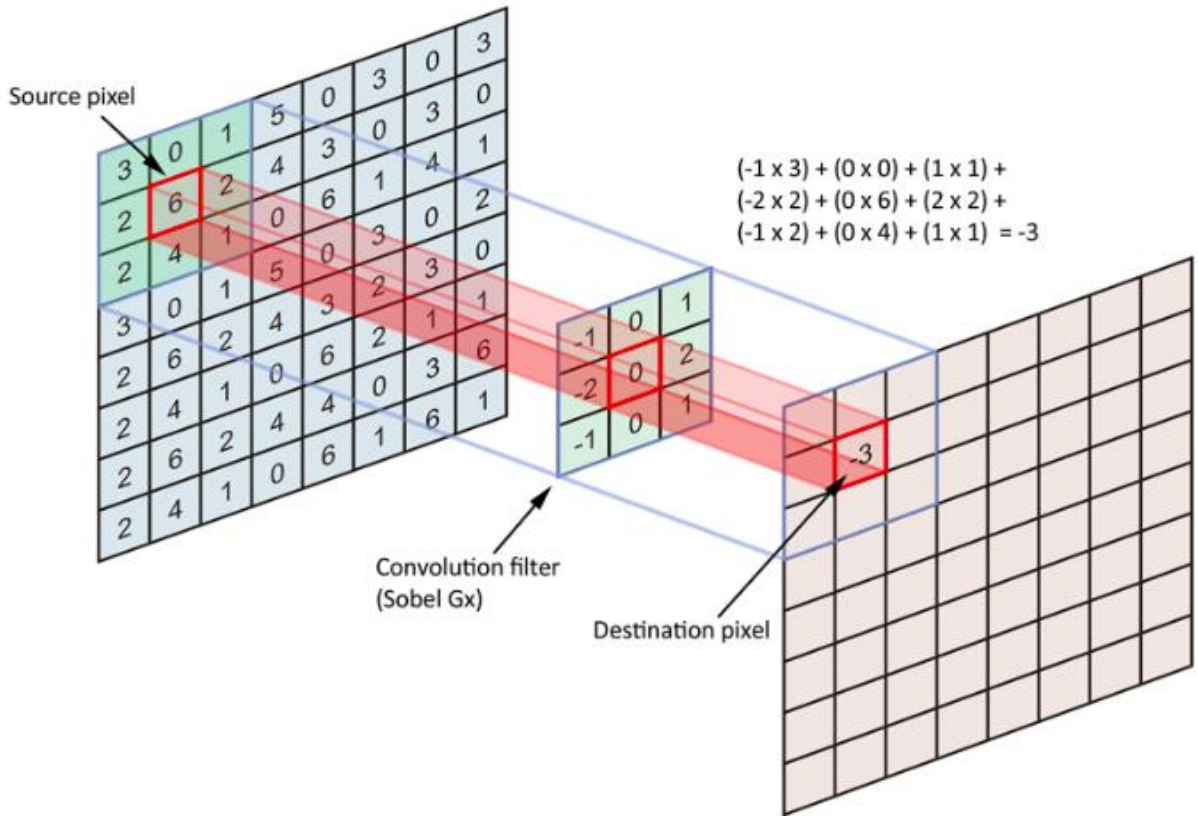


Figure 4.1 Convolutional Layer

Convolutional neural networks (CNNs)' basic building element is the convolutional layer. Convolution is carried out, which entails applying several learnable filters (sometimes referred to as kernels) to the input image (Figure 4.1). Each filter moves across the image while

performing multiplications and summations at the element level to create feature maps that show spatial linkages and local patterns. The input of a CNN is a tensor with shape $(number\ of\ inputs) \times (input\ height) \times (input\ width) \times (input\ channels)$. The image is abstracted into a feature map, also known as an activation map, after passing through a convolutional layer, and has the following shape$(number\ of\ inputs) \times (feature\ map\ height) \times (feature\ map\ width) \times (feature\ map\ channels)$.

- **ReLu Activation Function:**

CNNs frequently use the Rectified Linear Unit (ReLU) (Figure 4.2) activation function to add nonlinearity to the network. The highest value between zero and the input value is referred to as ReLU. It aids the network's learning of the data's intricate and asymmetrical relationships. The ReLU function is written as follows:

$$f(x) = x^+ = \max(0, x) = \frac{x + |x|}{2} = \begin{cases} x\ if\ x > 0, \\ 0\ otherwise. \end{cases} f'(x) = \begin{cases} 1\ if\ x > 0, \\ 0\ if\ x < 0. \end{cases}$$

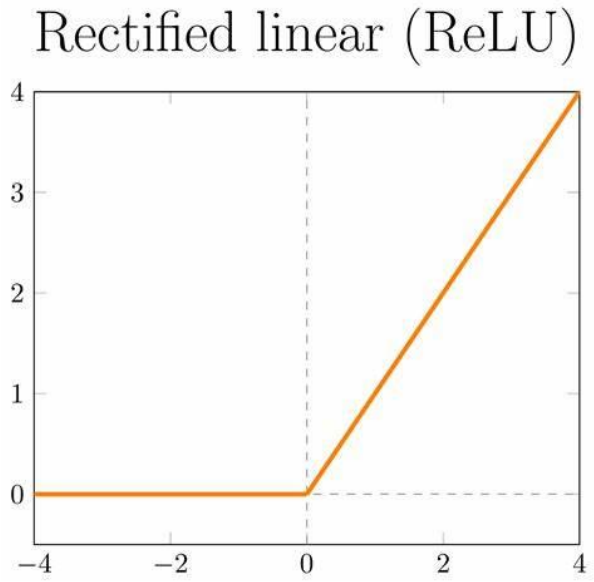


Figure 4.2 Rectified Linear Activation Function

- **Softmax Activation Function:**

When performing multi-class classification tasks, the output layer of a CNN frequently employs the Softmax activation function (Figure 4.3). It takes the raw output values from the preceding layer and normalizes them across all classes to create probabilities. The outputs can be interpreted as class probabilities because the Softmax function makes sure that the predicted probabilities add up to one. The standard Softmax function $\sigma: \mathbb{R}^K \to (0,1)^K$ is defined when $K \geq 1$ by the formula:

$$\sigma(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}\ for\ i = 1, \ldots, \boldsymbol{K}\ and\ \boldsymbol{z} = (z_1, \ldots, z_k) \in \mathbb{R}^K$$

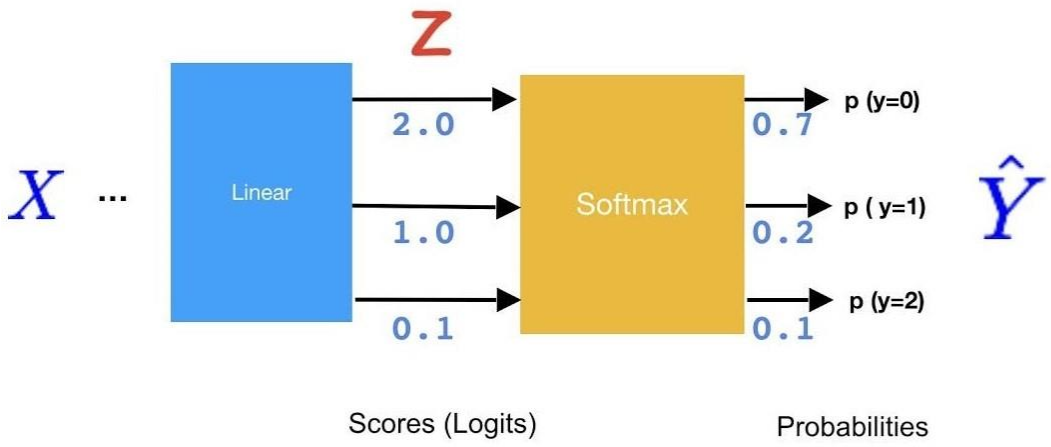


Figure 4.3 Softmax Activation Function

- **MaxPooling Layer**

The feature maps produced by the convolutional layers are downsampled using the MaxPooling layer. The most important data is preserved while the feature maps' spatial dimensions are reduced. The feature map is divided into non-overlapping regions via MaxPooling, and the highest value is chosen within each region. The network's computational complexity is decreased while the most dominating features are extracted using this method. The following is a representation of the MaxPooling (Figure 4.3):

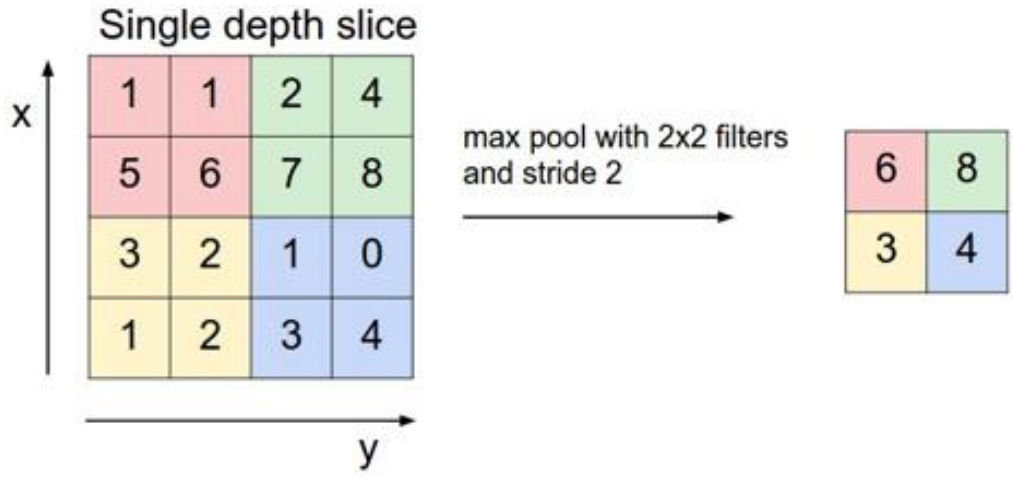


Figure 4.4 MaxPooling Layer

- **Dense Layer**

The final classification or regression work in a CNN is carried out by the dense layer, which is also referred to as the completely linked layer. Because every neuron in one layer is connected to every neuron in the layer above it, it makes it possible to learn complex relationships. A weighted sum of the inputs is computed and then passed through an activation function, such as ReLU or SoftMax, to produce the dense layer's output (Figure 4.5). The dense layer assists in converting the class probabilities or regression outputs from the high-level characteristics extracted from prior layers.

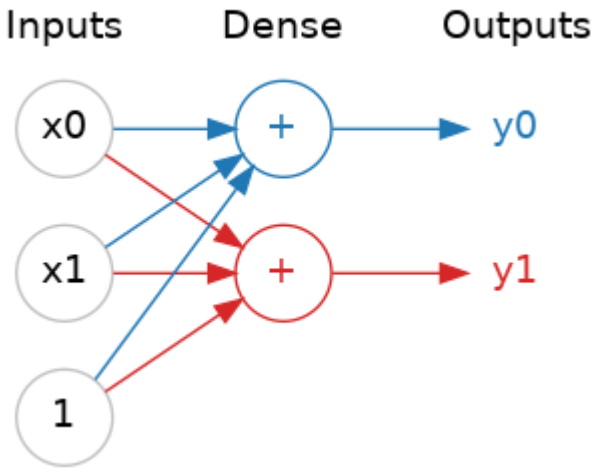


Figure 4.5 Dense Layer

Together, the convolutional layer, ReLU activation function, Maxpooling layer, Dense layer, and Softmax activation function enable the final classification, downsampling of the feature maps, extraction of meaningful features, introduction of non-linearity, and generation of class probabilities in CNNs. With the help of these components, CNN architectures can perform a variety of image processing tasks while learning intricate patterns and making precise predictions.

Forward propagation and backpropagation are two steps in the training process for CNNs designed to detect pneumonia. The input images are fed into the network during forward propagation, and the outputs are computed. The difference between the predicted and actual values is then measured by comparing the predicted outputs to the true labels using a loss function. Binary cross-entropy and sigmoid cross-entropy are frequent loss functions for binary classification tasks.

The gradients of the loss function with respect to the network parameters are computed using backpropagation. These gradients show the size and direction of the updates required to reduce the loss. These gradients are used by optimization methods like stochastic gradient descent (SGD) or Adam to repeatedly change the weights of the network, enhancing the model's performance. The CNN can learn the most discriminative features for precise pneumonia identification because to this repeated training method.

CNN algorithms can learn to recognize visual patterns connected to pneumonia by being trained on large datasets of annotated chest radiographs. Radiologists and other healthcare professionals would benefit greatly from the automated analysis and identification of pneumonia made possible by this machine learning method.

In conclusion, by utilizing their capacity to capture spatial hierarchies and learn discriminative features from medical images, convolutional neural networks have changed the field of pneumonia identification. CNNs may reliably categorize chest radiographs through iterative training, assisting in the early recognition and diagnosis of pneumonia. This example of machine learning in action shows how cutting-edge computational methods have the potential to enhance patient care and healthcare outcomes.

4.4. Model Architecture and Selection

In order to create a system that effectively detects pneumonia, the model architecture choice is crucial. When it comes to extracting intricate patterns and features from the input data, different architectures have differing capabilities. This chapter will look at three well-known deep learning architectures, ResNet50, InceptionV3, and VGG16, and thoroughly assess each one's performance in the context of pneumonia diagnosis.

The main selection criterion for models is accuracy. Based on chest X-ray pictures, the selected architecture should exhibit high accuracy when correctly identifying pneumonia and normal cases. For healthcare practitioners to receive trustworthy diagnostic information, forecasts must be accurate. Another important factor to consider is interpretability. The ability to comprehend and interpret the model's decision-making process is referred to as interpretability. In the medical industry, pneumonia detection systems that can explain patterns or emphasize characteristics that are important to their predictions are quite beneficial. Clinicians can check and comprehend the system's outputs thanks to interpretability, which also increases transparency and helps to foster confidence. In practical applications, computational effectiveness is important. Models with high computing requirements for training and inference may not be deployable in healthcare settings with limited resources. It is desirable to use effective models that balance computational demands with accuracy since they can be quickly implemented on edge devices or inside of already-existing healthcare infrastructure.

In evaluating the performance of the three architectures, we will employ rigorous experimental procedures. Our dataset will be split into training, validation, and test sets. During the training stage, the training set will be used to optimize the model parameters. We can fine-tune the hyperparameters and choose the top-performing model using the validation set. Finally, the final performance and generalization abilities of the chosen models will be evaluated using the testing set, which is hidden during the model construction process.

To assess the accuracy of the models, we will use conventional assessment measures such as accuracy, precision, recall, and F1 score. These measures give us important information about how well the models can differentiate between pneumonia and normal instances. The area under the receiver operating characteristic curve (AUC-ROC), which evaluates the trade-off between true positive rate and false positive rate at various classification thresholds, will also be considered.

Beyond accuracy, we will carefully examine the models' interpretability. The chest X-ray images will be visualized using the activation maps to highlight significant areas that contribute to the model's predictions. We can learn more about the characteristics and patterns that are suggestive of pneumonia by analyzing the decision-making process used by the model. By measuring the training and inference times, we will also assess the architectures' computational efficiency. For real-time application in clinical settings, models that can achieve high accuracy with minimal processing overhead are more likely to be practical.

We can choose the best architecture for pneumonia diagnosis by carefully comparing the capabilities of VGG16, ResNet50, and InceptionV3 in terms of accuracy, interpretability, and computational efficiency. The chosen architecture will be the starting point for our further

research and analysis, ultimately assisting in the creation of a precise, understandable, and computationally effective pneumonia detection system.

4.4.1. VGG16

A popular convolutional neural network design is VGG16, which Simonyan and Zisserman created in 2014 [24]. It has 16 layers, including 3 fully linked layers and 13 convolutional layers. The VGG16 network is renowned for its uniformity, simplicity, and modest 3x3 filter sizes. It has proven to be quite effective in a number of computer vision tasks.

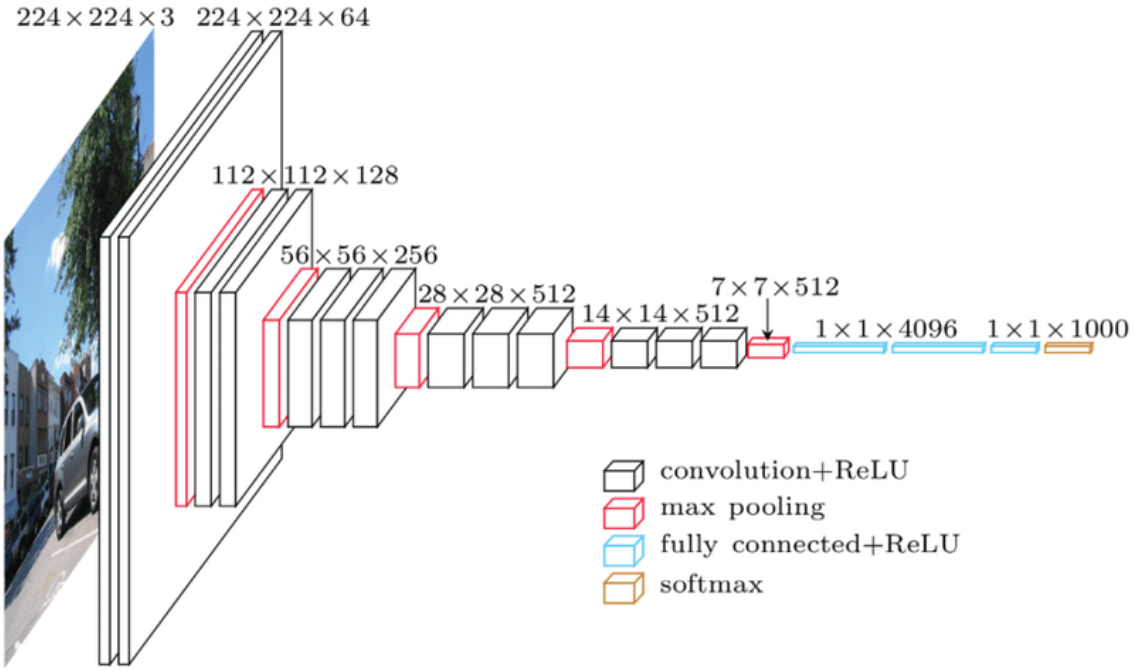


Figure 4.6 Architecture of VGG16

Max-pooling layers are added after a stack of convolutional layers in the VGG16 architecture in order to down sample the spatial dimensions. The classifier is comprised of the completely linked layers at the network's end. The biggest disadvantage of VGG16, despite its high accuracy, is its high parameter count, which makes training and using it computationally expensive.

4.4.2. ResNet50

A member of the ResNet family of architectures that tackles the vanishing gradient problem is ResNet50, which was unveiled by He et al. in 2016 [25]. A deep residual network with 50 layers is called ResNet50. By introducing skip connections, or shortcuts, it makes it possible to train incredibly deep networks by allowing gradients to flow directly through the network.

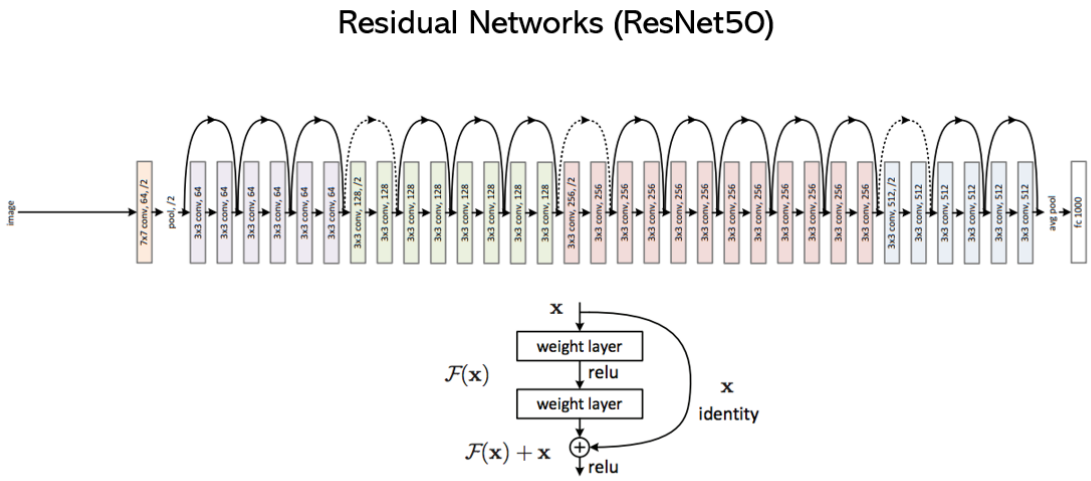


Figure 4.7 Architecture of ResNet152V2

Each residual block in the ResNet50 architecture contains many convolutional layers. Identity skip connections aid in information propagation throughout the network, enabling the training of deeper models without suffering performance deterioration. ResNet50 has proven to perform at the cutting edge of image classification problems.

4.4.3. InceptionV3

Another well-known convolutional neural network design is InceptionV3, which was created by Szegedy et al. in 2015 [26]. It is intended to determine the best way to balance computational efficiency with model complexity. Utilizing several parallel convolutional procedures with various kernel sizes, InceptionV3's inception modules capture characteristics at various scales.

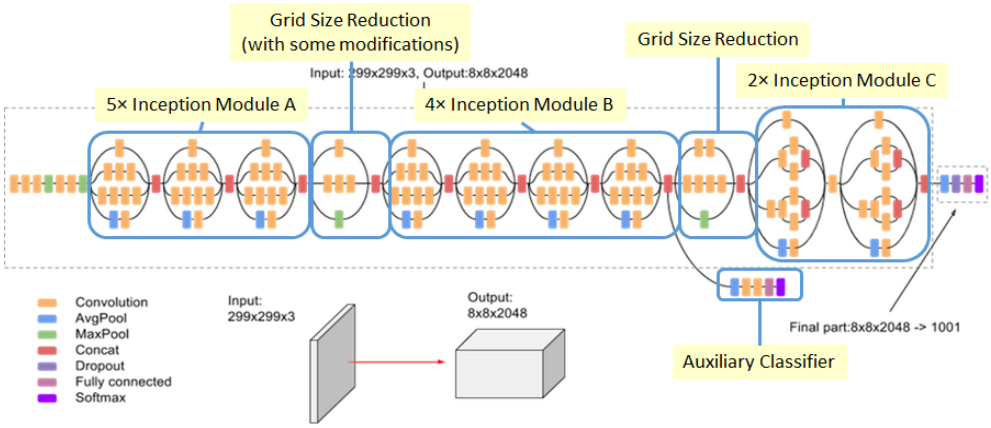


Figure 4.8 Architecture of DenseNet121

A succession of inception modules is followed by fully connected layers for categorization in the InceptionV3 architecture. The model can learn accurate representations of complicated patterns because to these modules' feature extraction at various sizes and resolutions. Model complexity and computational performance are well-balanced in InceptionV3, making it appropriate for contexts with limited resources.

4.4.4. Model Selection

The performance and results of the system can be considerably impacted by choosing the best model architecture for pneumonia detection. The method of selecting amongst the VGG16, ResNet50, and InceptionV3 architectures based on the evaluation metrics given in the preceding chapter will be covered in this subsection. The detection of pneumonia cases and the trade-off between false positives and false negatives will receive particular attention.

It is critical to consider the assessment metrics that offer information about the model's performance when choosing a model design. We put a lot of emphasis on accuracy, precision, recall, and F1 score when it comes to pneumonia identification.

The **accuracy** of a model's predictions is measured by considering both true positives and true negatives. For unbalanced datasets, where one class (in this case, normal cases) may be underrepresented, accuracy alone may not be the best metric.

Precision is the percentage of accurate positive predictions among all the model's positive predictions. Precision in the context of pneumonia detection refers to a model's capacity to accurately detect pneumonia cases among the expected positives. As it reduces false positives, a high precision number is preferred.

Recall, also referred to as sensitivity or the true positive rate, quantifies the percentage of real positive cases that the model accurately recognized. Recall is the capacity to accurately identify pneumonia cases among all the real pneumonia patients in the setting of pneumonia detection. Recall must be increased to reduce false negatives.

The **F1 score** is a balanced indicator of the model's performance because it is the harmonic mean of precision and recall. For comparing models and making judgments based on their overall performance, it integrates precision and recall into one metric.

These metrics will be looked at more detail in the subchapter of Evaluation Metrics.

The goal of pneumonia detection is to reliably identify cases of the disease, even if that results in more false positives. Prioritizing recall is essential since missed pneumonia cases (false negatives) can have detrimental effects on patient health. False positives should still be considered because they could result in unnecessary follow-up tests or treatments, but the focus should be on reducing false negatives. Early detection of pneumonia cases can aid in prompt treatment initiation and stop the disease's progression, improving patient outcomes.

We can choose the model architecture after carefully considering the assessment criteria and the emphasis on identifying pneumonia patients.

Each of the three models—VGG16, ResNet50, and InceptionV3—has advantages and disadvantages that were covered in the previous subchapter. Although VGG16 produces good accuracy, it is computationally expensive due to its numerous parameters. Deeper models may be trained thanks to ResNet50, which solves the vanishing gradient issue. ResNet50 successfully balances complexity with effectiveness.

It is advised to use a model architecture that can produce higher precision even if it leads to a little lower accuracy given the emphasis on recognizing pneumonia patients and

prioritizing precision. Precision is essential for reducing false negatives and ensuring accurate diagnosis of pneumonia cases.

When analyzing the performance measures and giving the precision metric priority, ResNet50 performs better than both VGG16 and InceptionV3. In comparison to the other models, ResNet50 exhibits greater precision values and fewer false negatives.

The model's deep residual learning architecture, which enables the network to successfully handle the vanishing gradient problem, is ResNet50's primary strength. ResNet50's skip connections make it easier for data to move around the network, enabling it to acquire and store crucial attributes required for pneumonia identification.

ResNet50 can efficiently learn complicated representations and discern minute details in chest X-ray pictures thanks to its deep layer structure. With fewer false negatives, it can effectively detect pneumonia patients, which contributes to its remarkable recall performance.

ResNet50 is the suggested model architecture in this study for accurate and quick pneumonia detection due to the emphasis on memory and the capacity to reduce missed or misclassified pneumonia cases.

In later chapters, we will examine the specifics of how the selected models were implemented, the metrics they offered, and a comparison of their strengths and weaknesses.

4.5. Federated Learning Approach

4.5.1. Overview of Federated Learning

A decentralized machine learning paradigm called federated learning enables model training on numerous distributed data sources while maintaining data privacy and confidentiality. This chapter will provide an overview of the federated learning strategy we'll use to detect pneumonia using the Flower federated learning library. We will go into the model aggregation procedure as well as client-server communication.

Collaborative model training is made possible by federated learning without the necessity for data centralized in one place. Instead, the model is locally trained on each client device or edge node that is responsible for holding the relevant data. The updates to the models are then collected on a main server, which manages the training procedure. As the client data stays on their devices and is not shared with the server or other clients, this decentralized approach assures privacy and data security.

Learning a single, global statistical model from data kept on tens to maybe millions of remote devices is the standard federated learning challenge. We intend to learn this model under the restriction that device-generated data is locally stored and analyzed, with only intermediate updates being routinely exchanged with a central server. The objective is often to reduce the following objective function in particular:

$$min_{\omega}\, F(\omega), \qquad where\ F(\omega) \coloneqq \sum_{k=1}^{m} p_k F_k(\omega) \quad (1)$$

In this case, *m* is the number of devices, $p_k \geq 0$ and $\sum_k p_k = 1$, and F_k is the local objective function for the *k*th device. The local objective function over is sometimes described as the empirical risk over local data, i.e. $F_k(\omega) = \frac{1}{n_k}\sum_{j_k=1}^{n_k} f_{jk}\left(\omega; x_{jk}, y_{jk}\right)$, where $n_k$is the quantity of locally available samples. With two default values of $p_k = \frac{1}{n}$ or $p_k = \frac{n_k}{n}$, where $n = \sum_k n_k$ is the total number of samples, the user-defined term p_k describes the relative impact of each device [27].

In practice, by doing a few rounds of local optimization, each participant typically acquires and improves a global consensus model before exchanging updates, either directly or through a parameter server. The likelihood that the entire approach is minimizing (Eq. 1) decreases as more rounds of local training are conducted. The actual method for aggregating features depends on the topology of the network because nodes may be divided into subnetworks because of geographical or legal restrictions (see Fig. 3.3). Aggregation techniques might rely on several nodes without any centralization or on a single aggregation node. An example is peer-to-peer FL, where connections exist between all, or a portion of participants and model updates are exchanged solely between directly connected sites. In contrast, Algorithm 1 provides an example of centralized FL aggregation. A client may decide to share only a portion of model parameters in order to reduce communication costs, improve privacy protection or create multi-task learning algorithms with only a portion of their parameters learned in a federated manner. Aggregation strategies do not necessarily require information about the full model update [28].

Algorithm 1. Example of a FL algorithm [29] via Hub & Spoke (Centralized topology) with FedAvg aggregation

Require: num_federate_rounds T

procedure AGGREGATING
Initialize global model $W^{(0)}$
for t ← 1 … T **do**
  for client k ← 1 … K **do** ▸ *Run in parallel*
    Send $W^{(k-1)}$ to client k
    Receive model updates and number of local trainings iterations. $\left(\Delta W_k^{(t-1)}, N_k\right)$ from client's local training with $F_k\left(X_k; W^{(t-1)}\right)$
  end for
  $$W^{(t)} \leftarrow W^{(t-1)} + \frac{1}{\sum_k N_k}\sum_k \left(N_k - W_k^{(t-1)}\right)$$
  end for
return $W^{(t)}$
end procedure

4.5.2. Communication between Clients and Server

For coordination, model updates, and model aggregation in a federated learning system, good communication between the clients and the server is essential. To enable coordinated training and aggregation of the global model, communication requires the exchange of data, model parameters, and instructions. We'll make use of the gRPC (Google Remote Procedure

Call) framework to make this communication easier. A high-performance, open-source framework called gRPC makes it possible for clients and servers to communicate effectively and flexibly. It is excellent for real-time and iterative model updates in federated learning because it enables bidirectional streaming and employs the Protocol Buffers (protobuf) data serialization method to specify the message structure.

For client-server communication in federated learning, gRPC has a number of benefits:

- **Efficient and Scalable Communication:**

The HTTP/2 protocol, which allows for multiplexing and asynchronous processing, is used by gRPC. This increases communication efficiency by enabling the processing of several requests and responses at once. As a result, gRPC can effectively manage large, federated learning systems with several clients.

- **Bidirectional Streaming:**

Iterative updates between the clients and the server are a feature of federated learning. The server can deliver model commands and parameters to clients while also receiving model updates from them thanks to gRPC's bidirectional streaming. This two-way data exchange speeds up federated learning and allows for real-time cooperation.

- **Strong Message Serialization:**

The message structures sent between the clients and the server are defined by gRPC using Protocol Buffers, a binary serialization standard independent of a language. We can define the message format using a clear and extensible syntax with Protocol Buffers. Data consistency and interoperability between various programming languages used by clients and the server are guaranteed by Protocol Buffers' strong type.

- **Interoperability and Language Support:**

Python, Java, C++, and many other programming languages are supported by gRPC. This enables the federated learning system's flexibility and interoperability by allowing clients and the server to be implemented in many languages. Additionally, to make the implementation process simpler, gRPC generates client and server code depending on the specified protobuf message format.

- **Security and Authentication:**

Transport Layer Security (TLS) encryption is supported natively by gRPC, ensuring secure communication between clients and servers. This aids in preserving the confidentiality and accuracy of the data sent during federated learning. To confirm the identities of the clients and server, gRPC also provides authentication mechanisms such token-based authentication or mutual TLS authentication.

A client-server architecture governs communication between clients and servers using the gRPC framework. The server serves as the federated learning process' central coordinator, starting communication and overseeing it. Clients sign up with the server, which enables it to keep track of and manage their involvement.

The server instructs the clients on the local model training process during the federated learning process by sending model parameters and instructions (Fig 4.4.1). These instructions are given to the clients, who then use the appropriate datasets to execute local training. They produce model updates that reflect the adjustments made to the local model during training, which are frequently expressed as gradients or differential parameters.

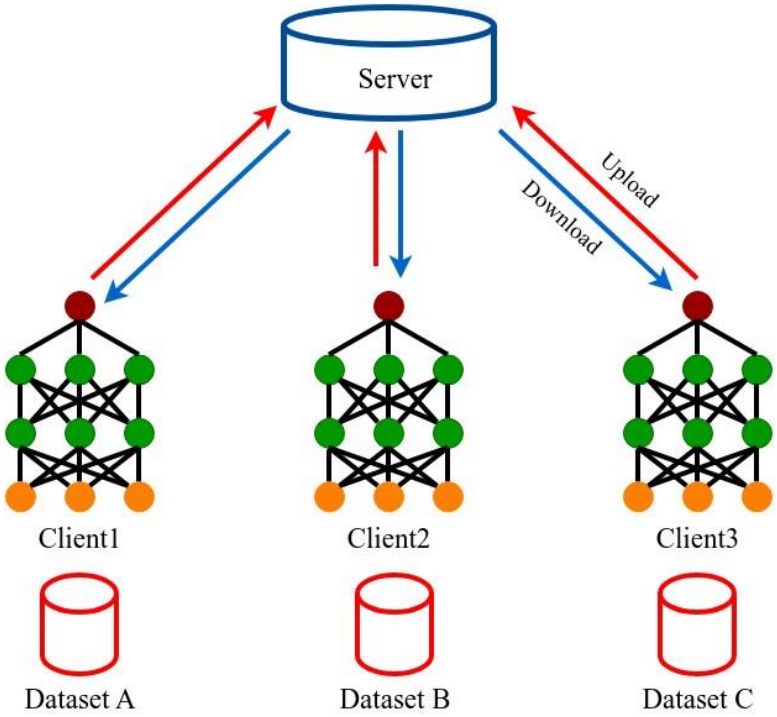


Figure 4.9 Communication between Clients and Server

Real-time model updates are sent from the clients to the server using gRPC. These updates are delivered to the server, which then aggregates them to create a new global model. The customers are subsequently given access to the global model that has been combined for the subsequent training cycle.

The federated learning system collectively enhances the performance of the global model while preserving data privacy and security through the iterative process of communication, local training, update creation, and aggregation.

In summary, the gRPC architecture supports scalable and effective client-server communication in federated learning. It supports message serialization, strong typing, and bidirectional streaming, enabling real-time interaction and compatibility with many programming languages. The security elements that are already included in gRPC enable secure and authenticated communication, safeguarding the confidentiality of the sent data. We can efficiently coordinate and synchronize the training and aggregation operations in our federated learning system for pneumonia detection by utilizing gRPC.

4.5.3. Model Aggregation

A crucial process in federated learning is called model aggregation, in which the server compiles model changes from many clients to create an updated global model. The goal of the aggregation process is to make use of the clients' combined expertise while protecting the confidentiality of individual data sources.

Typical methods for model aggregation include:

- **Weighted Averaging:**

In this method, the weights for the clients' model updates are determined by the server based on predetermined criteria like the quantity of local training samples or the model's performance with the local data. The updates are averaged, and updates from clients who have more trustworthy or representative data are given more weight.

- **Federated Averaging:**

Federated Learning employs the well-liked aggregation method known as federated averaging. It entails averaging the model updates the clients have provided. The global model is updated to reflect the participants' combined knowledge by adding together all model changes and dividing the amount by the number of clients.

- **Secure Aggregation:**

During the aggregation process, secure aggregation approaches seek to improve privacy and security. These methods make use of cryptographic protocols to maintain the confidentiality of the individual model updates while enabling aggregate operations on the server. Methods for secure aggregation guard against possible data leaks from model updates.

In our situation, Federated Averaging (FedAvg) was chosen. To aggregate the model updates, the Federated Averaging algorithm follows a step-by-step process. Consider a K-client federated learning system. The algorithm works in the following way:

Algorithm 2. FederatedAveraging. The K clients are indexed by k; B is the local minibatch size, E is the number of local epoch and η is the learning rate.

Server executes:

procedure AGGREGATING
initlialize w_0
for each round t = 1, 2 … **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow (random\ set\ of\ m\ clients)$ // *in our case 2 clients*
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow ClientUpdate(k, w_t)$
  end for
  $m_t \leftarrow \sum_{k \in S_t} n_k$
  $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$
end for

ClientUpdate(*k*, *w*)**:** // *Run on client k*
12: $B \leftarrow (split\ P_k\ into\ batches\ of\ size$ B)
13: **for** each local epoch I from 1 to E **do**
14: **for** batch $b \in B$ **do**
15: $w \leftarrow w - n\nabla l(w; b)$
16: **end for**

Let's use a graph to represent Federated Averaging so that it is easier to understand. Consider our scenario of simulating pneumonia detection using federated learning with two clients:

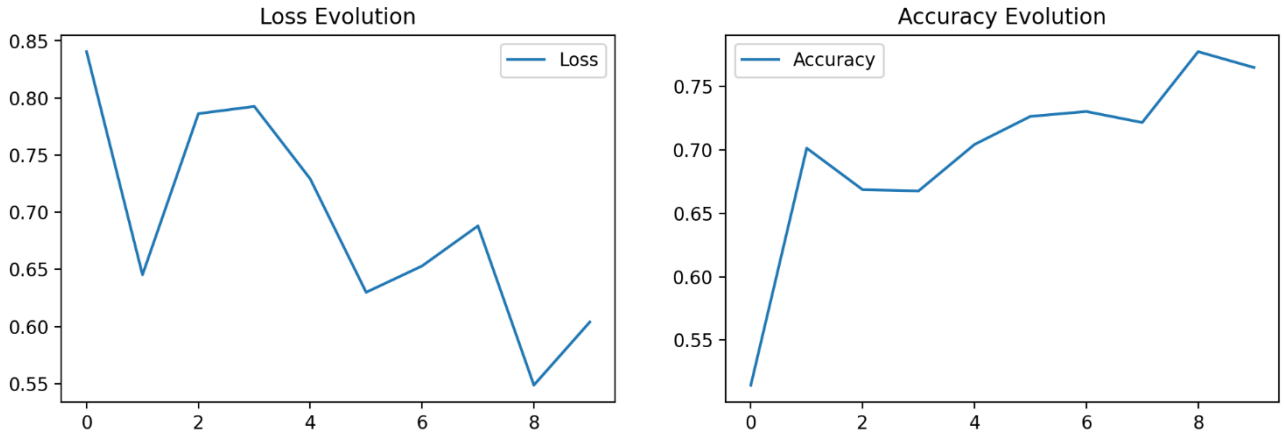


Figure 4.10 Evolution of the loss and accuracy of the model based on training rounds.

The number of communication rounds is represented on the x-axis in the graph above, while the loss (left) and accuracy (right) of the global model is shown on the y-axis. The global model's performance is shown by the blue line as it gets better with each iteration.

The clients create model changes and conduct local training during each round. Federated Averaging is used to aggregate the updates, and the global model is then adjusted accordingly. As the rounds go on, the global model improves as a result of incorporating the insights from all involved clients.

The Federated Averaging method averages each client's contributions to balance them out. With this strategy, the aggregation process is ensured to be fair and not to be dominated by any one client's update. The global model gains from the many viewpoints and local facts included in the client datasets by aggregating the updates.

It is important to note that the convergence and effectiveness of the global model are strongly influenced by the learning rate (n) and the quantity of communication rounds. To balance the influence of the aggregated updates and avoid overfitting or underfitting, a suitable learning rate should be selected. The ideal number of rounds must be carefully chosen because too few rounds can lead to a global model that is not fully developed while too many rounds can cause overfitting or excessive communication overhead.

In our instance, we chose 10 training rounds, the first of which had one epoch for connection testing and the subsequent nine had 40 epochs. We can see that the global model converges at round number five in the picture above (Fig. 4.4.2), thus achieving the highest accuracy and lowest loss. This analysis confirms Federated Averaging's ability to improve the pneumonia detection model by utilizing the collective expertise of the participating clients.

In conclusion, Federated Learning's model aggregation method of choice is Federated Averaging. It enables the global model to gain from the collective knowledge while protecting data privacy by averaging the model updates produced by many clients. The convergence and efficiency of Federated Averaging in improving the pneumonia detection model are revealed through visualization, analysis, and performance evaluation.

4.5.4. Evaluation Metrics

The effectiveness of the pneumonia detection model can be evaluated in large part by looking at evaluation metrics. We may evaluate the model's efficacy and reach conclusions about its performance by quantifying several features of its predictions. We will examine frequently used evaluation metrics like accuracy, loss, precision, recall, and F1 score in this chapter.

- **Accuracy**

A key evaluation statistic, accuracy assesses the general accuracy of the model's forecasts. Out of all the examples in the evaluation dataset, it represents the percentage of instances that were correctly categorized. The equation for accuracy is:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Here, TP (True Positives) denotes the number of pneumonia cases that were correctly predicted, TN (True Negatives) the number of non-pneumonia cases that were correctly predicted, FP (False Positives) the number of non-pneumonia cases that were incorrectly classified as pneumonia, and FN (False Negatives) the number of pneumonia cases that were incorrectly classified as pneumonia.

High accuracy means that both pneumonia and non-pneumonia cases are correctly predicted by the model. But accuracy by itself could not give a full picture, particularly in datasets with imbalances or when the cost of misclassification varies between classes.

- **Loss**

Loss is a metric measuring the difference between expected and actual values. It displays the degree of error in the predictions made by the model. Binary cross-entropy loss or mean squared error are two common loss functions for tasks involving the identification of pneumonia.

For each training instance, the loss is calculated and utilized to update the model's parameters during the optimization process. A smaller loss shows that the model's predictions

are more in line with reality. Monitoring the loss during training enables evaluation of the model's convergence and optimization development.

We utilize binary cross-entropy because in our scenario, the model only distinguishes between the presence or absence of pneumonia based on two characteristics. Binary cross-entropy loss can be described using the following equation:

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \cdot log\hat{Y}_i + (1 - Y_i) \cdot \log(1 - \hat{Y}_i))$$

Here, Y_i represents the true label of the instance (either 0 or 1), and $\hat{Y}_i$ represents the predicted probability of the positive class (i.e., the probability of the instance being classified as pneumonia). The logarithm function is applied to the predicted probability and its complement to calculate the loss.

- **Precision**

Precision is an assessment metric that gauges how accurately the model predicts the future. It displays the percentage of actual pneumonia cases among all pneumonia cases that were previously suspected. The formula for precision is:

$$Precision = \frac{TP}{TP + FP}$$

When accurately identifying instances of pneumonia is essential, precision is especially helpful, and false positives should be kept to a minimum. A high precision score indicates that the model has a low rate of misclassification of instances that are not pneumonia.

- **Recall**

The capacity of the model to properly detect positive occurrences (pneumonia patients) is measured by recall, also known as sensitivity or true positive rate. It shows the percentage of actual pneumonia cases that the model properly recognized. Recall is determined by:

$$Recall = \frac{TP}{TP + FN}$$

False negatives should be kept to a minimum and recall is crucial when identifying cases of pneumonia. The model can successfully identify the majority of the pneumonia cases in the dataset if the recall value is high.

- **F1 Score**

The F1 score strikes a balance between precision and recall by combining both into a single statistic. It is calculated as follows to determine the harmonic mean of recall and precision:

$$F1\ Score = 2 \cdot \frac{Precision\ \cdot Recall}{Precision + Recall}$$

The F1 score has a range of 0 to 1, with 1 representing flawless recall and precision. In situations where we wish to consider both false positives and false negatives, it is a helpful statistic.

The precise criteria and goals of the pneumonia detection system determine the choice of evaluation metrics. Precision, recall, and the F1 score offer information on class-specific performance, whereas accuracy is a frequently used indicator to evaluate overall performance.

Understanding the model's advantages and disadvantages depends on how the evaluation measures are interpreted. For instance, a high accuracy but low recall indicates that the model could have trouble detecting every incidence of pneumonia. A high recall but low precision, on the other hand, denotes a higher rate of false positives. When understanding the evaluation metrics, it is critical to take the context and requirements of the pneumonia detection task into account. It is possible to evaluate the model's effectiveness, spot potential improvement areas, and make well-informed judgments about the model's deployment and optimization by thoroughly analyzing these indicators.

In conclusion, evaluation parameters including accuracy, loss, precision, recall, and F1 score offer crucial information about how well the pneumonia detection model performs. We may evaluate the model's accuracy, error, class-specific performance, and overall efficacy in identifying pneumonia patients by considering a variety of variables.

Chapter 5. Detailed Design and Implementation

5.1. Python and the Libraries Used

Python is the primary language used in data science and machine learning and is a flexible programming language. It was important in the creation of the technique for detecting pneumonia. Python offers a vast range of modules and frameworks that significantly streamline activities like data manipulation, visualization, machine learning, and deep learning. This project utilized a variety of significant libraries, each of which served a specific purpose.

For managing multi-dimensional arrays, Numpy, a foundational library for numerical computing in Python, offered effective data structures and functions. It made it possible to do operations like matrix manipulation and calculations, which were crucial for preparing and analyzing the dataset.

Data exploration and manipulation were greatly aided by the potent data analysis library Pandas. It included practical data structures, like DataFrames, that made it easier to do things like import and preprocess the dataset, handle missing values, and do statistical analysis.

Insightful plots and graphs were produced using the well-known data visualization packages Seaborn and Matplotlib. They made it possible to visualize different statistical distributions, the connections between variables, and performance indicators. Gaining a deeper grasp of the dataset and model performance was made easier by these representations.

The pneumonia detection model was constructed and trained using TensorFlow, a popular deep learning toolkit. TensorFlow's user-friendly and effective API made it possible to build deep neural networks, use unique loss functions, and apply optimization methods. Additionally, it offered resources for computing performance indicators and evaluating models.

The extensive machine learning library Scikit-learn was used for tasks including data partitioning, model evaluation, and performance metric calculation. It offered a variety of machine learning tools and algorithms, such as cross-validation, hyperparameter tuning, and model selection help.

Finally, the distributed training process was implemented using Flower Federated Learning, a federated learning library. By facilitating the coordination of training across several clients while protecting data privacy, Flower makes it easier to construct federated learning algorithms. It is a good option for collaborative training on dispersed data sources because it abstracts away the difficulties of communication, aggregation, and synchronization between clients and the server.

These libraries were chosen because of their broad functionality, strong community backing, and well-established track record in the industry. By utilizing these potent tools, it was possible to construct a strong and successful pneumonia detection system by streamlining processes, ensuring the adoption of recognized best practices, and facilitating efficient development.

5.2. Overview of the Dataset

The "Chest X-Ray Images (Pneumonia)" dataset provided by Paul Mooney was the source of the data utilized for the pneumonia detection algorithm, which was obtained via Kaggle. This dataset is an important tool for developing and testing pneumonia detection methods. There are numerous chest X-ray photos in it that are either labeled "Normal" or "Pneumonia." From retrospective cohorts of children patients aged one to five at the Guangzhou Women and Children's Medical Center in Guangzhou, chest X-ray images were chosen. All chest X-ray imaging was done as part of the regular clinical treatment provided to patients [30].

The dataset consists of 5856 chest X-ray images in total, 1583 of which have the label "Normal," and 4273 of which have the label "Pneumonia." The pictures are from several sources and feature X-rays with various image quality, patient statistics, and pneumonia symptoms. Due to the dataset's diversity, pneumonia detection models that can handle a range of clinical practice settings can be trained and evaluated.

The dataset is split into three subsets: training, validation, and test sets to ensure the robustness and generalizability of the pneumonia detection model. This split enables accurate model performance assessment and aids in avoiding overfitting.

The gradient descent method is used to train the model's parameters on the training set, which is the largest subset. It gives the necessary information for the model to pick up from different pneumonia and regular chest X-ray images and learn and generalize. The training set consists of 4192 images, from which 1082 are labeled as “Normal” and 3110 images as “Pneumonia”.

The model's performance during training is evaluated using the validation set, and choices for hyperparameter tuning and model selection are made based on this evaluation. It acts as an objective evaluation set that directs the model's fine-tuning and helps avoid overfitting. The validation set consists of 1038 images, from which 266 are labeled as “Normal” and 772 as “Pneumonia”.

The test set will use the remaining space in the dataset. On this set, the trained model's ultimate performance on unobserved data is assessed. We can get a fair assessment of the model's performance and establish how well it detects pneumonia by testing it on a different, independent test set. The test set consists of 626 images, from which 235 are labeled as “Normal” and 391 as “Pneumonia”.

The dataset's separation into training, validation, and test sets enables a thorough assessment of the model's performance and guarantees its capacity to generalize to fresh, unexplored data. To guarantee that the model is trained and evaluated on a balanced and diverse dataset, care is taken to preserve a representative distribution of pneumonia and normal patients in each subgroup. The final distribution of the dataset can be observed in the following table:

Table 5.1. Distribution of the Dataset

	Total	**Normal**	**Pneumonia**	**Percentage (%)**
Train	4192	1082	3110	~72%
Validation	1038	266	772	~18%
Test	626	235	391	~10%

Let's take a closer look at a few examples photographs to have a better idea of the dataset. The chest X-ray images in the "Normal" category show sound lungs free of pneumonia symptoms. These images show distinct lung architecture and clean lung fields.

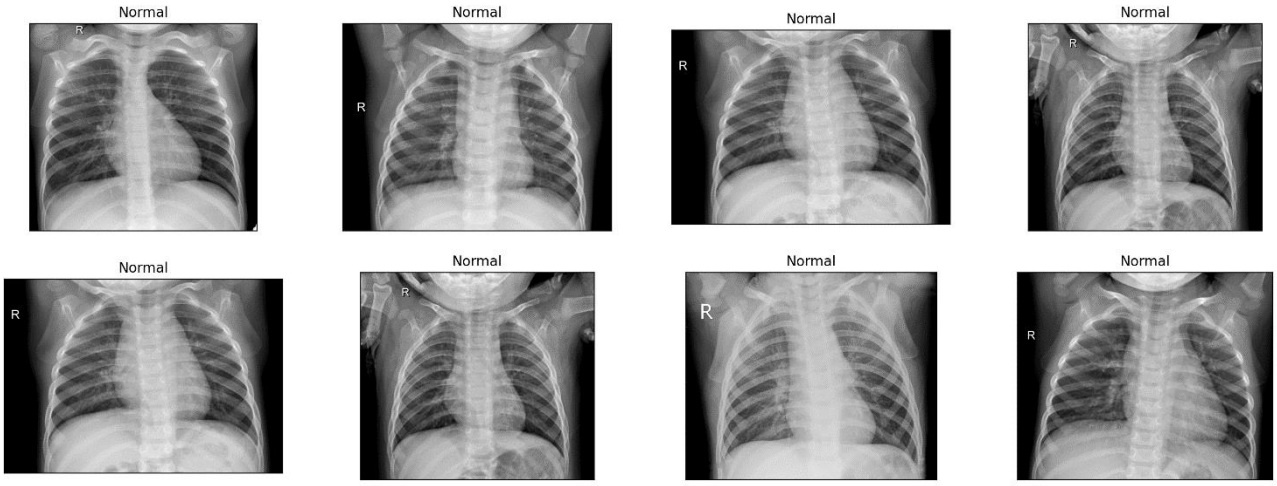


Figure 5.1 Sample Normal Chest X-Rays

On the other side, the "Pneumonia" category contains X-ray pictures of people who have been given the diagnosis of pneumonia. These images display a variety of pneumonia-related abnormalities, including opacities, infiltrates, and consolidations in the lung fields.

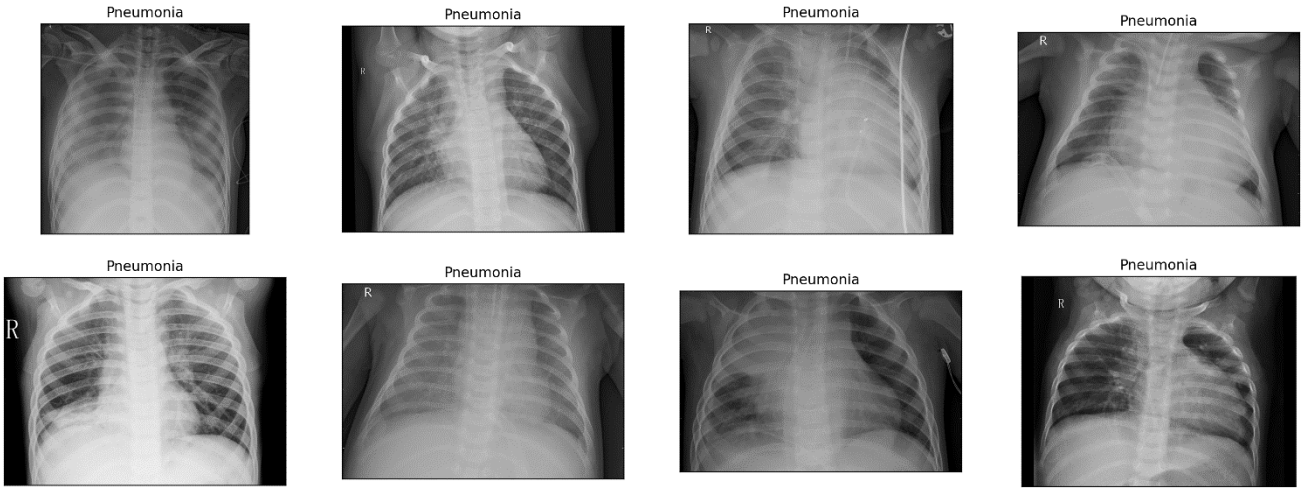


Figure 5.2 Sample Pneumonia Chest X-Rays

The photos in the pneumonia dataset have various sizes, which correspond to various chest X-ray sizes. These measurements, which are commonly expressed in pixels, include the images' width and height. To ensure compatibility with the selected model architecture, it is crucial to take the image dimensions into account.

Additionally, comprehension of the dataset's pixel values offers insights regarding the image intensity range. In an 8-bit image, the grayscale intensity of each pixel is represented by its pixel value, which can vary from 0 (black) to 255 (white). We can establish the dynamic range of the dataset by looking at the minimum and maximum pixel values, which aids in the preprocessing and normalizing processes.

A measurement of the overall brightness of the images can also be obtained by computing the mean value of the pixel intensities. The dataset may be visualized using this information, and any potential biases in the distribution of the images can be found. The degree of variation or contrast present in the photographs is also shown by the standard deviation of the pixel values.

We learn crucial details about the properties of the pneumonia dataset by examining the dimensions, pixel values, mean, and standard deviation of the photographs. The resizing, normalization, and augmentation operations required to get the data ready for training the pneumonia detection model are guided by these statistics.

For our dataset, we can see that images typically have a width of 1858 pixels and a height of 2090 pixels with a single-color channel. The highest possible pixel value is 255, and the lowest is 0. The pixels' average value is 128.9075, with a standard deviation of 62.3010. All these can be observed in Figure 5.3.

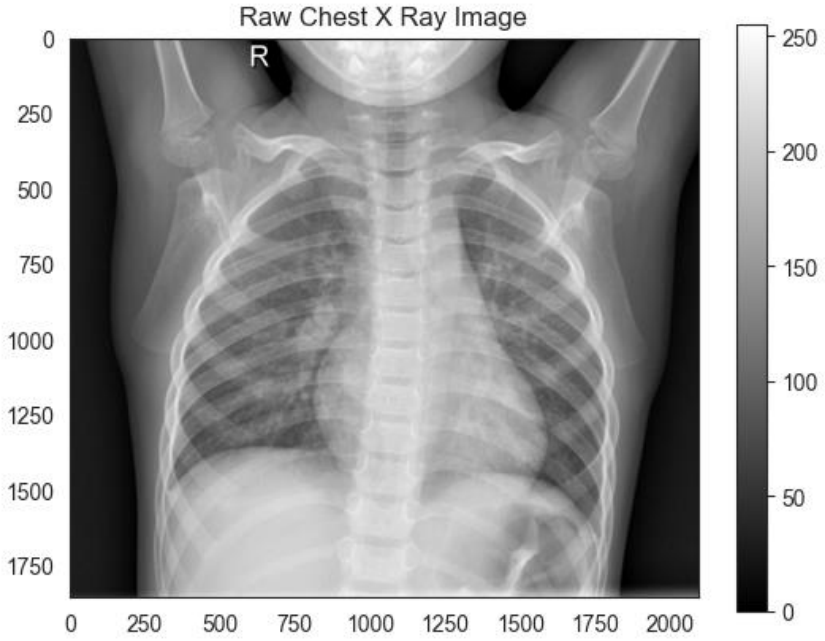


Figure 5.3 Analysis of Chest X-Ray

It is significant to note that a group of skilled radiologists and medical experts carefully analyzed and annotated the information. The labels are accurate and trustworthy thanks to their experience, which qualifies the dataset for machine learning model training.

For detecting pneumonia, the "Chest X-Ray Images (Pneumonia)" dataset from Kaggle offers a large and varied collection of chest X-ray images. Its size, variety, and meticulously labeled photos make it an invaluable tool for developing and testing pneumonia detection models. The image preprocessing methods used to get the dataset ready for training the pneumonia detection model will be covered in detail in the following chapter.

5.3. Image Preprocessing

An essential step in developing a reliable and effective pneumonia detection model is image preprocessing. The Keras ImageDataGenerator class was used in this project to perform data augmentation and apply different preprocessing methods. This class offers an easy approach to quickly create enhanced images on-the-fly, enabling more diverse training data and enhancing the generalization skills of the model.

Algorithm 3. ImageDataGenerator for data augmentation.

```
1:  image_generator = ImageDataGenerator(
2:      rotation_range=20,
3:      width_shift_range=0.1,
4:      shear_range=0.1,
5:      zoom_range=0.1,
6:      samplewise_center=True,
7:      samplewise_std_normalization=True
8:  )
```

Several augmentation settings were set for the ImageDataGenerator to add variability to the training data. These included a rotation range (line 2) to rotate the images randomly, a width shift range (line 3) to shift the images horizontally, a shear range (line 4) to apply shearing transformations, and a zoom range (line 5) to zoom into the images arbitrarily. The model became more resistant to various orientations, locations, and scales of pneumonia-related characteristics because of the changes these augmentations helped create in the training data.

Each image in the training data also underwent samplewise centering (line 6) and standard deviation normalization (line 7). By removing the mean pixel value from each image, samplewise centering successfully centers the data around zero. To ensure that the pixel values have the same scale, standard deviation normalization divides the centered images by their standard deviation. These preprocessing techniques assist in data normalization and increase the stability of the training process.

Algorithm 4. Generators for train, validation and test.

```
1:  train = image_generator.flow_from_directory(train_dir,
2:                                              batch_size=8,
3:                                              shuffle=True,
4:                                              class_mode='categorical',
5:                                              target_size=(180, 180))
6:
7:   validation = image_generator.flow_from_directory(val_dir,
8:                                                   batch_size=1,
9:                                                   shuffle=False,
10:                                                  class_mode='categorical',
11:                                                  target_size=(180, 180))
12:
13:  test = image_generator.flow_from_directory(test_dir,
14:                                             batch_size=1,
15:                                             shuffle=False,
16:                                             class_mode='categorical',
17:                                             target_size=(180, 180))
```

Separate data generators for the training, validation, and test sets were made after the augmentation and preprocessing settings had been established. A batch size of 8 (line 2) was selected for the training generator to process 8 photos at once. To incorporate randomization into the training process, the photos were randomly jumbled (line 3). To maintain consistency in the neural network's input size, the target size of the images was fixed at 180x180 pixels (line 5). To tackle the multi-class classification problem where the target labels are one-hot encoded, the class mode was set to "categorical" (line 4).

Similar but somewhat different settings were used for the validation and test generators. To process one image at a time while validating and testing, the batch size was set to 1 (line 8, 13), because in real-life scenario images are not processed a batch at a time and knowing the average per batch would effectively give the model an advantage. To preserve the original arrangement of the validation and test sets, the photos were not randomized (line 9, 15). This made it possible to compare and evaluate the model's performance on hypothetical data in a consistent manner.

The training, validation, and test data were successfully created for training the pneumonia detection model by using the ImageDataGenerator and specifying the proper augmentation and preprocessing settings. The augmented images are shown in Figure 5.4. These methods enhanced the training data's diversity, enhanced the model's capacity to generalize to new cases, and guaranteed evaluation process consistency.

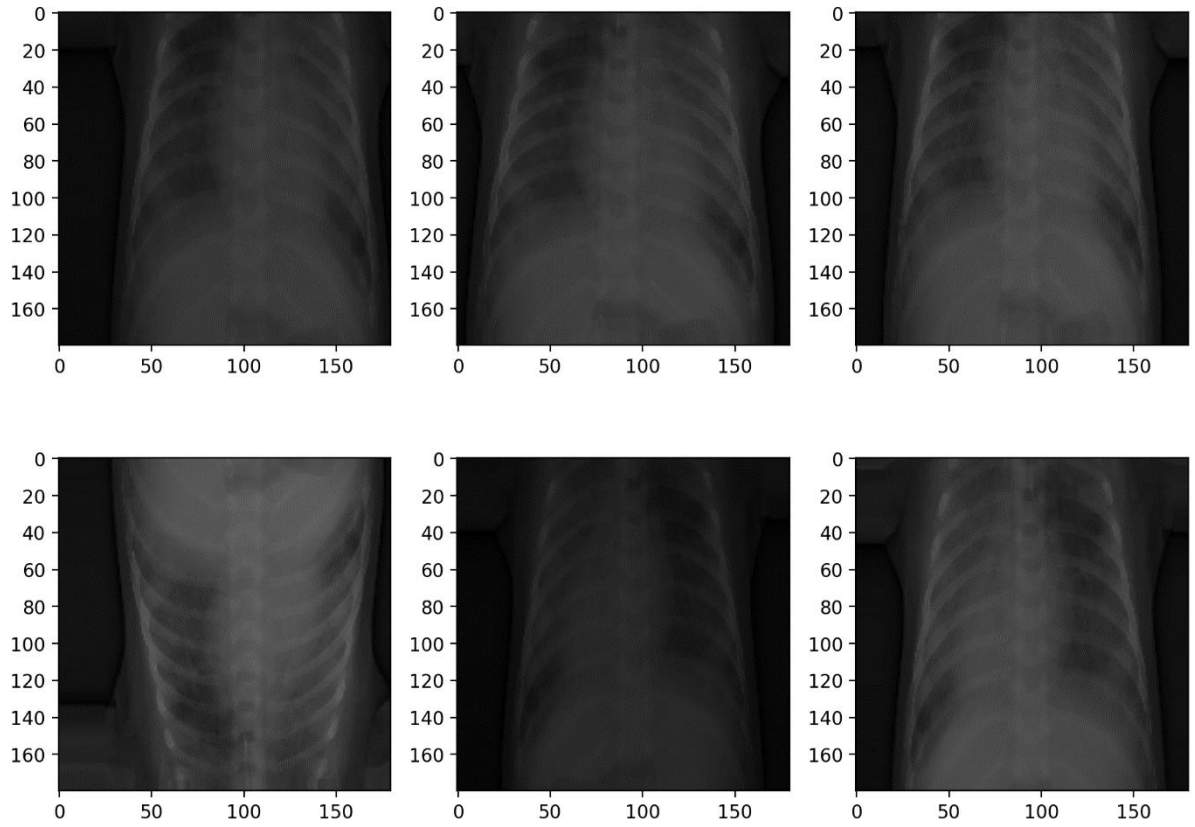


Figure 5.4 Augmented Images of Chest X-Rays

5.4. Transfer Learning

Transfer learning is a strong deep learning technique that uses previously taught models to speed up and enhance the training of new tasks. In this section, we compare the effectiveness of three well-known pre-trained models for detecting pneumonia: VGG16, ResNet50, and InceptionV3. For each model, we added additional layers to the pre-trained base network and fine-tuned the model for pneumonia classification.

Algorithm 4. VGG16 model architecture

```
base = VGG16(weights = 'imagenet',
                    include_top = False,
                    input_shape = (180, 180, 3))

for layer in base.layers:
    layer.trainable = False

vgg_model = Sequential()
vgg_model.add(base)
vgg_model.add(GlobalAveragePooling2D())
vgg_model.add(BatchNormalization())
vgg_model.add(Dense(256, activation='relu'))
```

```
13:  vgg_model.add(Dropout(0.5))
14:  vgg_model.add(BatchNormalization())
15:  vgg_model.add(Dense(128, activation='relu'))
16:  vgg_model.add(Dropout(0.5))
17:  vgg_model.add(Dense(2, activation='softmax'))
18:
19:  optm = Adam(lr=0.0001)
20:  vgg_model.compile(loss='categorical_crossentropy', optimizer=optm,
21:                metrics=['accuracy'])
```

For the final layers of each model included in this comparison, I employed the same architecture; the model's foundation layer was the only one that was different.

First, the base model is specified (lines 1-3), and uses the model's pre-trained weights from its training on the ImageNet dataset (line 1). As additional classification layers are added and the input shape of the images that are given to the model is specified (line 3), the fully linked layers at the top of the model are next disabled (line 2).

The ‘for’ loop iterates trough each layer of the ‘base’ model and sets them as non-trainable (line 5-6). These layers' weights will be frozen so that only the weights of the new categorization layers will be learned during training.

Next a model ‘Sequential’ model is created with the ‘base’ being the one discussed before (line 8-9). To decrease the features' spatial dimensions and produce a feature vector with a fixed length, a global average pooling layer is implemented (line 10). In order to improve training stability, a batch normalization layer is introduced to normalize the activations of the preceding layer (line 11). ReLU activation and a fully connected layer with 256 units are introduced (line 12). Overfitting is decreased by adding a dropout layer that randomly deactivates 50% of the neurons during training (line 13). Another batch normalization layer and dropout layer are added and finally the output is added to the last dense layer (line 17), which has 2 units (the two cases ‘Normal’ and ‘Pneumonia’) and a Softmax activation function.

After the model architecture is defined, it is also compiled with a specific optimizer and a loss function. The optimizer used is Adaptive Moment Estimation (Adam) with a learning rate of 0.0001 (line 19). For gradient-based optimization methods, the Adam optimizer is a well-liked option, and the weight update process's step size is determined by the learning rate. The loss function is set to categorical cross-entropy (line 10), which is typically used for multi-class classification problems. It calculates the cross-entropy between the true labels and the predicted probabilities. Finally, 'accuracy' is the metric that is used to assess the model during training and testing (line 21). This metric counts how many samples out of all the samples were correctly categorized.

By switching the VGG16() function from line 1 to ResNet50() or InceptionV3(), the model base layer is changed from VGG16 to ResNet50 and InceptionV3.

Convolutional layers from the pre-trained models are combined with extra fully connected layers for classification in this model architecture. The additional layers learn to transfer these features to the required output classes while the pre-trained layers serve as feature

extractors. By compiling the model with these settings, you are ready to train and evaluate the transfer learning-based model using the specified optimizer, loss function, and metrics.

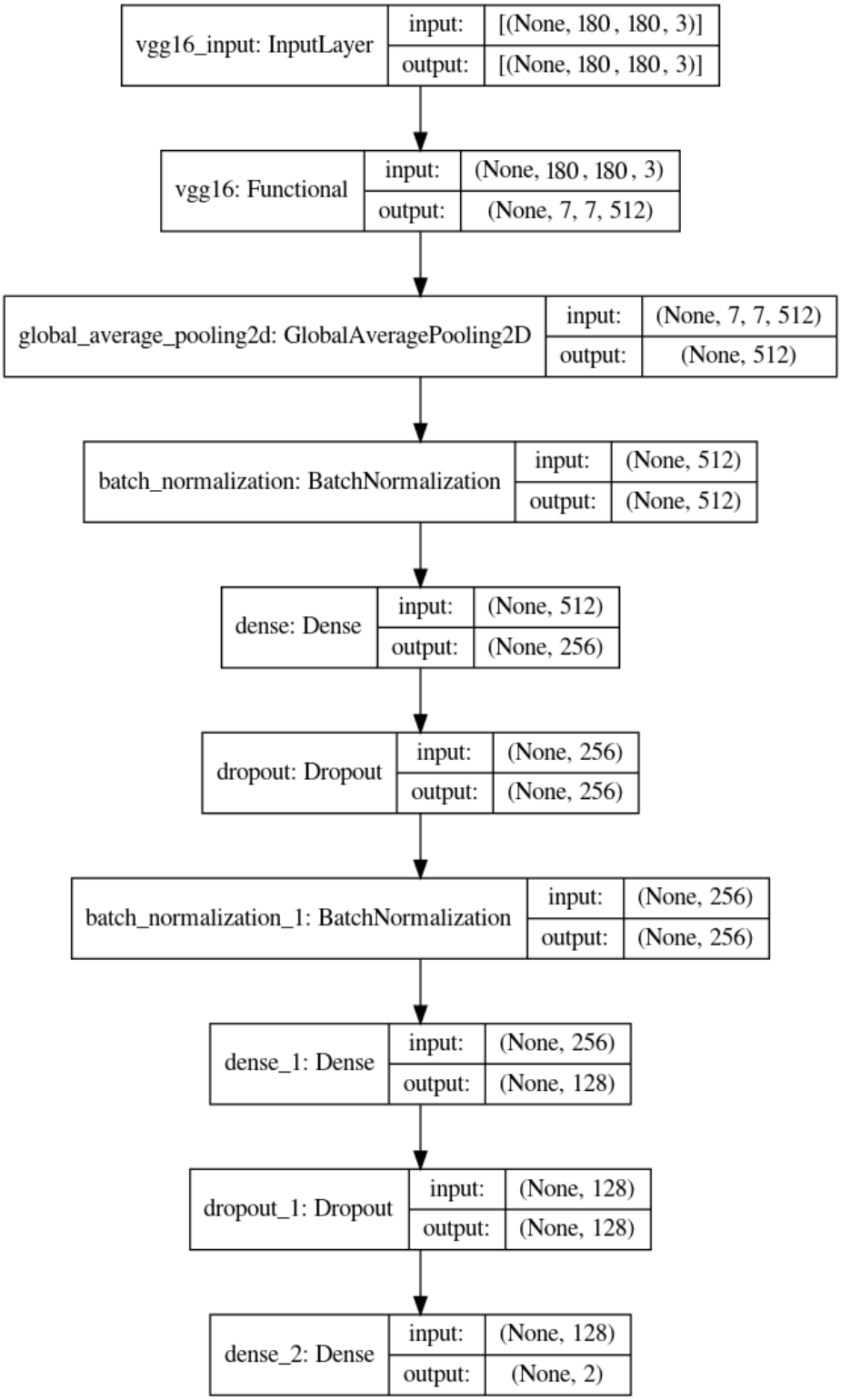


Figure 5.5 Model architecture based on VGG16

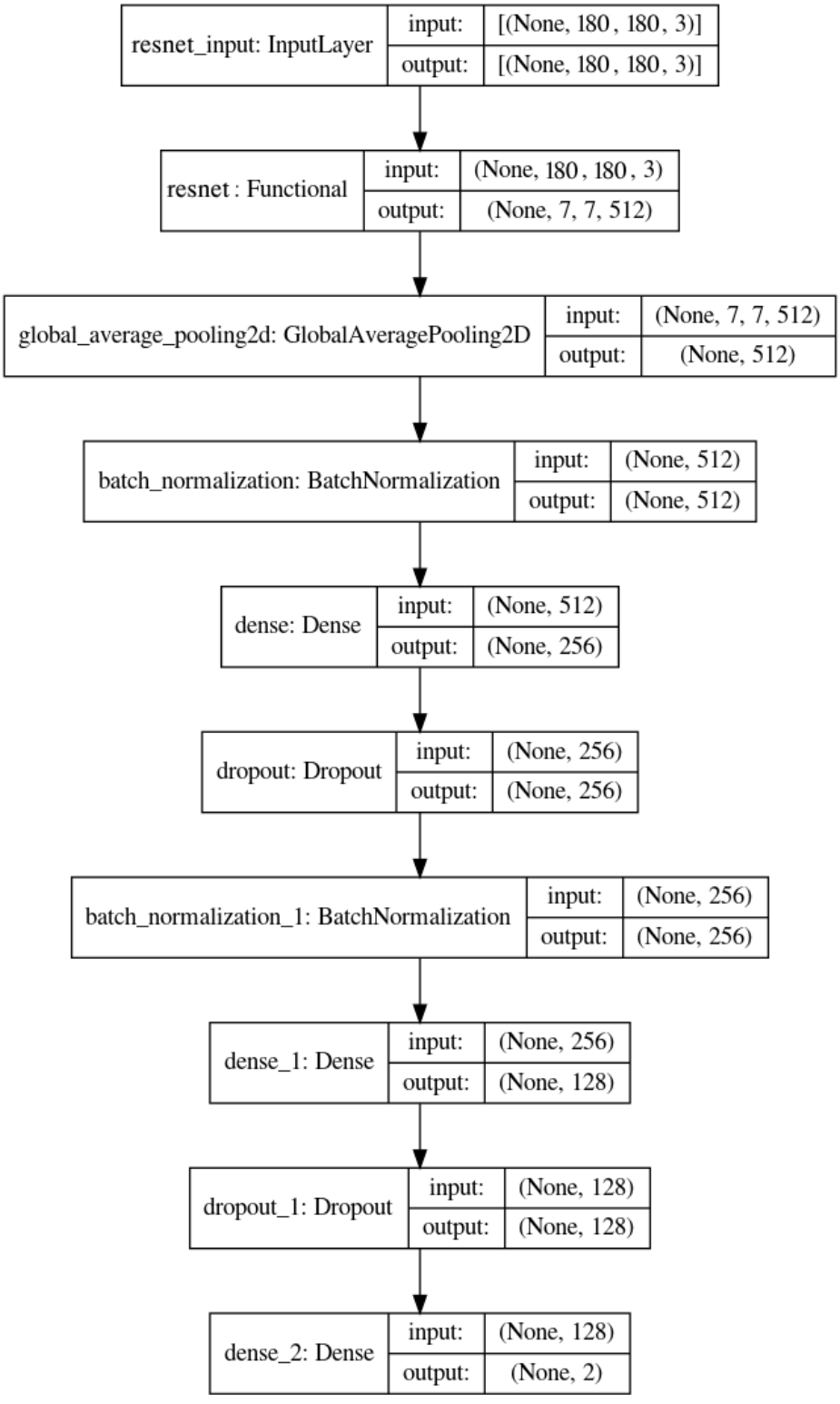


Figure 5.6 Model architecture based on ResNet50

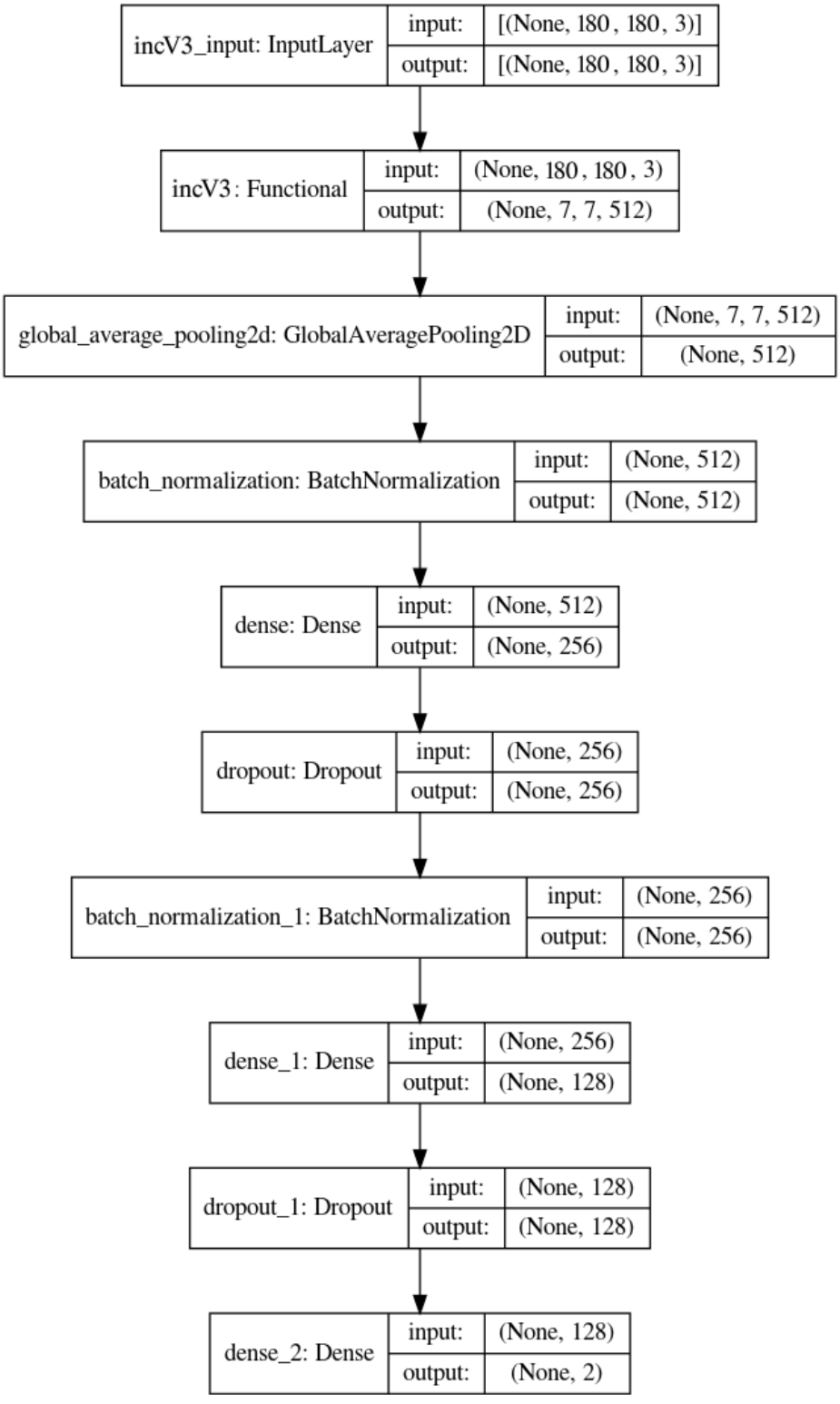


Figure 5.7 Model architecture with InceptionV3

A federated learning network with two clients and one server was used to test the models. The dataset was split equally between the two clients in this design, giving each client access to half of the data (2096 images). Using their respective data, the clients trained their local models, which were then updated and provided to the server for aggregation.

After the training phase, the model's evaluation was carried out server-side. The server gathered the client's model updates and created a global model from them. The performance on a different validation dataset that wasn't used during training was then assessed using this global model. After that, another test set is used to access the global model achieved after the last round of training. The metrics will be presented in the chapter “Testing and Validation”.

5.5. Federated Learning Implementation

Compared to conventional centralized methods, federated learning has several advantages. It enables businesses to train machine learning models on private or sensitive data without disclosing that information to a centralized server. Federated learning is especially well suited for situations where data cannot be easily consolidated due to privacy rules or proprietary concerns because of the decentralized nature that guarantees data privacy and security.

A comprehensive and user-friendly framework for federated learning implementation is offered by the Flower library. It is a fantastic option for creating federated learning systems because it has many benefits.

Firstly, Flower offers a high-level abstraction layer that makes it easier to develop federated learning algorithms. Developers can concentrate on the fundamental logic of their models and training procedures since it provides a simple and understandable API that abstracts away the challenges of distributed training. This abstraction layer dramatically lowers the entrance hurdle for federated learning and allows for quicker testing and development.

Secondly, Flower offers extensive support for a variety of federated learning algorithms and techniques. Flower offers the required building blocks and tools, whether you need to construct Federated Averaging (FedAvg), Federated Proximal Gradient Descent (FedProx), or other cutting-edge algorithms. Researchers and practitioners can investigate various federated learning strategies and customize them for their own use cases thanks to this flexibility.

The scalability and effectiveness of Flower are other noteworthy qualities. It is made to manage complex federated learning scenarios involving a huge number of users. To reduce communication overhead and boost overall training effectiveness, Flower uses strategies like client sampling, client weighting, and communication compression. Flower can tackle federated learning tasks in real-world environments that involve a variety of gadgets and data sources thanks to its scalability.

Furthermore, Flower offers strong support for a variety of frameworks and platforms. Users can take advantage of their current models and training pipelines by integrating it smoothly with well-known deep learning frameworks like TensorFlow and PyTorch. To allow federated learning in distributed contexts, Flower also supports several deployment methods, including local clusters, cloud infrastructure, and edge devices.

Flower also encourages knowledge sharing and open collaboration. It is an open-source project with a vibrant community that encourages creativity and ongoing development. Flower

is a dependable and secure option for deploying federated learning systems because of the community-driven development that makes sure it keeps up with the most recent developments in federated learning research and industry practices.

We must set up a server and numerous clients to implement federated learning using the Flower library. The server serves as the training process's central coordinator, collecting model changes from clients and overseeing the training process itself. On the other hand, the clients hold their own data and carry out local model training using that data.

The training set will be used with the augmentations discussed in the previous chapters. This set is split into 2 equally sized subsets (2096 images) and distributed to two clients that will be used locally for training the models. Also, on the server side, the validation set (1038 images) is used for evaluating each round of training and the test set (626 images) is used for the final evaluation with the converged model (Figure 5.8).

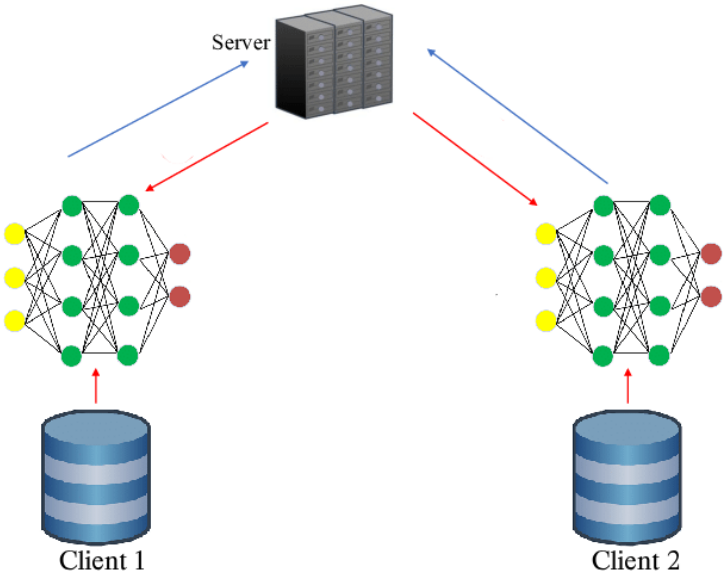


Figure 5.8 Pneumonia Detection System Architecture

Next the model InceptionV3 that was chosen in the previous chapter is initialized. Because there are for training 541 images of normal chest X-rays and 1555 images of pneumonia chest X-rays the following algorithm will be used to correct the imbalance between these classes:

Algorithm 5. Class weight

```
1:  weight_for_0 = num_total / (2 * num_normal)
2:  weight_fot_1 = num_total / (2 * num_pneumonia)
3:
4:  class_weight = {0: weight_for_0, 1: weight_for_1}
```

As a result, the class imbalance between the normal and pneumonia chest X-rays is balanced out at 1.94 and 0.67, respectively.

Through the Client interface, the Flower server communicates with clients. The server broadcasts training instructions over the network when it decides which client to train. After receiving those directives, the client invokes one of the Client methods to execute your code.

When using Keras and Tensorflow, Flower offers a convenience class called "NumPyClient", that makes it simpler to implement the Client interface.

Algorithm 6. Client-side Implementation

```
1:  class PneumoniaClient(fl.client.NumPyClient):
2:      def get_parameters(self, config):
3:          return model.get_weights()
4:
5:      def fit(self, parameters, config):
6:          model.set_weights(parameters)
7:
8:          hyper_epochs: int = config["local_epochs"]
9:          model.fit(training_set, epochs=hyper_epochs)
10:         return model.get_weights(), len(x_train), {}
11:
12:     fl.client.start_numpy_client(server_address="server:50051",
13: client=PneumoniaClient())
```

The current model parameters are retrieved from the global server by the "get_parameters" method (line 2). It returns the weights of the model using the "model.get_weights()" function (line 3). The fit method is used for training the client's local model (line 5). It takes the received parameters from the server and sets them in the local model using "model.set_weights(parameters)" (line 6). The client then performs training on its local training dataset, specified as "training_set", for the number of epochs received from the server (line 8). After training, the updated weights are retrieved using model.get_weights() and returned along with the number of samples in the training set and an empty dictionary (line 10). With the help of the "start_numpy_client" function (line 12-13), a connection is made between the client and the server at the given address (server:50051). The Flower server's IP address and port number should be used as the server address. The client will interact with the server after it has been launched to take part in the federated learning process. Model parameters will be received from the server, local training or evaluation depending on the applied techniques will be performed, and then the updated model parameters will be sent back to the server for aggregation. The NumPy-based Flower client will connect to the designated server by running this code, which will assist in the collaborative training and evaluation of the pneumonia detection model in the federated learning environment. This is repeated for the second client, with the only difference being the dataset offered.

By coordinating the aggregation of model updates from the participating clients and carrying out evaluations, the server component of the application plays a significant function in federated learning. It performs the function of the central authority in charge of maintaining the global model and merging the locally trained models.

During the federated learning rounds, the server receives the updated model parameters from each client in the context of pneumonia detection. The server integrates the model updates

from the clients to produce a new global model using FedAvg (Federated Averaging). FedAvg is chosen over alternative aggregation techniques because the data is evenly divided across the clients, creating a balance between them, and eliminating the need to calculate client-specific weights.

The server can assess the global model after the aggregate is finished. It evaluates the model's performance on a different validation or test dataset using the evaluation mechanism set up in the server portion of the application. This assessment sheds light on the global model's overall accuracy and efficacy in diagnosing pneumonia.

The server allows the iterative improvement of the global model throughout the federated learning process by carrying out model aggregation and evaluations. It makes sure that the collective wisdom of the participating clients is utilized to improve the pneumonia detection model's generality and accuracy.

Algorithm 7. SaveModelStrategy

```
1:  class SaveModelStrategy(fl.server.strategy.FedAvg):
2:      def aggregate_fit(
3:          self,
4:          rnd: int,
5:          results: List[Tuple[fl.server.client_proxy.ClientProxy,
6:  fl.common.FitRes]],
7:          failures: List[BaseException],
8:      ) -> Tuple[fl.common.Parameters, Dict[str, Any]]:
9:          aggregated_parameters, aggregated_metrics = super().aggregate_fit(rnd,
10: results, failures)
11:         weights, metrics = super().aggregate_fit(rnd, results, failures)
12:         weights_list = fl.common.parameters_to_ndarrays(weights)
13:         if weights is not None:
14:             print(f"Saving round {rnd} weights...")
15:             model.set_weights(weights_list)
16:             model.save(f"model_round_{rnd}.h5")
17:         return weights, metrics
```

A custom strategy class is defined (line 1) which adds the option to save the model weights at the conclusion of each cycle of aggregation, expanding the usefulness of Federated Averaging. The code first calls the parent class's “aggregate_fit” function in the overridden “aggregate_fit” method to carry out the required aggregation and receive the aggregated parameters and metrics (line 9). Next, the “parameters_to_ndarrays” function from the “fl.common” module is used to turn the aggregated parameters into a list of numpy arrays (line 12). To access the model's individual weights, this step is required. The method then saves the weights using the “model.save” if the weights are not None, indicating that the aggregate was successful (line 15). The round number is included in the filename for identification when saving the weights and the extension of the file being “h5”, it refers to HDF5 (Hierarchical Data Format version 5), which is the most common way to save the weights of the model for Keras and Tensorflow.

Algorithm 8. Functions for Evaluation and Configuration

```
1:  def get_evaluate_fn(model):
2:      val_gen = ImageDataGenerator(rescale = 1./255)
3:      val_set = test_gen.flow_from_directory('val',
4:                                  target_size = (180,
5:                                                 180),
6:                                  batch_size = 1,
7:                                  class_mode = 'categorical')
8:
9:  def evaluate(
10:         server_round: int,
11:         parameters: fl.common.NDArrays,
12:         config: Dict[str, fl.common.Scalar],
13:     ) -> Optional[Tuple[float, Dict[str, fl.common.Scalar]]]:
14:         model.set_weights(parameters)
15:         loss, accuracy = model.evaluate(val_set)
16:         return loss, {"accuracy": accuracy}
17:     return evaluate
18:
19: def fit_config(server_round: int):
20:     config = {
21:         "local_epochs": 10
22:     }
23:     return config
```

The “get_evaluation_fn” function (line 1) creates the validation set used to evaluate the model after each round, is discussed in the previous chapters. The server round number, model parameters, and a configuration dictionary are all inputs to the evaluate function. Using “model.set_weights()” (line 14), the weights of the model are updated within the function using the supplied parameters. The model's accuracy and loss are then determined on the validation set “val_set” using the “model.evaluate()” method. The evaluation results are returned as a tuple, which includes the loss and a dictionary that contains the accuracy metric. A configuration dictionary particular to that round is returned by the “fit_config()” function (line 19), which accepts the server round number as input. The number of epochs supplied to the clients for their training is the sole key-value pair in this case's configuration dictionary. Model evaluation and configuration parameters are provided by these functions, which are intended to be utilized in a federated learning framework.

Algorithm 8. Server-side Implementation

```
1:      strategy = SaveModelStrategy(
2:          evaluate_fn=get_evaluate_fn(model),
3:          on_fit_config_fn=fit_config,
4:          initial_parameters=fl.common.ndarrays_to_parameters(
5:                                  model.get_weights()),
```

```
6:      )
7:
8:      fl.server.start_server(
9:          server_address="0.0.0.0:50051",
10:         config=fl.server.ServerConfig(num_rounds=10),
11:         strategy=strategy,
12:     )
```

A strategy is created using the “SaveModelStrategy” and the functions previously defined (line 1-6). This strategy is then used in the “fl.server.start_server()” method (line 8), which starts the federated learning process. The number of rounds of communication is set at 10 and the server address is set at “0.0.0.0:50051”.

A single computer is used to run the simulation, with two different command windows acting as the client and the server, respectively. This configuration enables the simulation of a distributed setting with client-server communication in a federated learning system. The clients and the server can communicate and work together during the training process since they are separately started in their respective command windows. This simulation demonstrates the capacity of the clients to train locally on their data and contribute to the aggregation of a global model on the server, all within the boundaries of a single machine, and allows for the evaluation of the efficacy of the federated learning approach.

Instead of using the GPU in this simulation, the CPU's computational capabilities are used. While using the CPU for the simulation has various benefits, GPUs are more frequently known for their great performance capabilities in deep learning applications. Having two machine learning processes on a single computer considerably benefits from the CPU because CPUs are better at multi-threaded applications.

The results of the federated training system will be discussed in the “Testing and Validation” chapter.

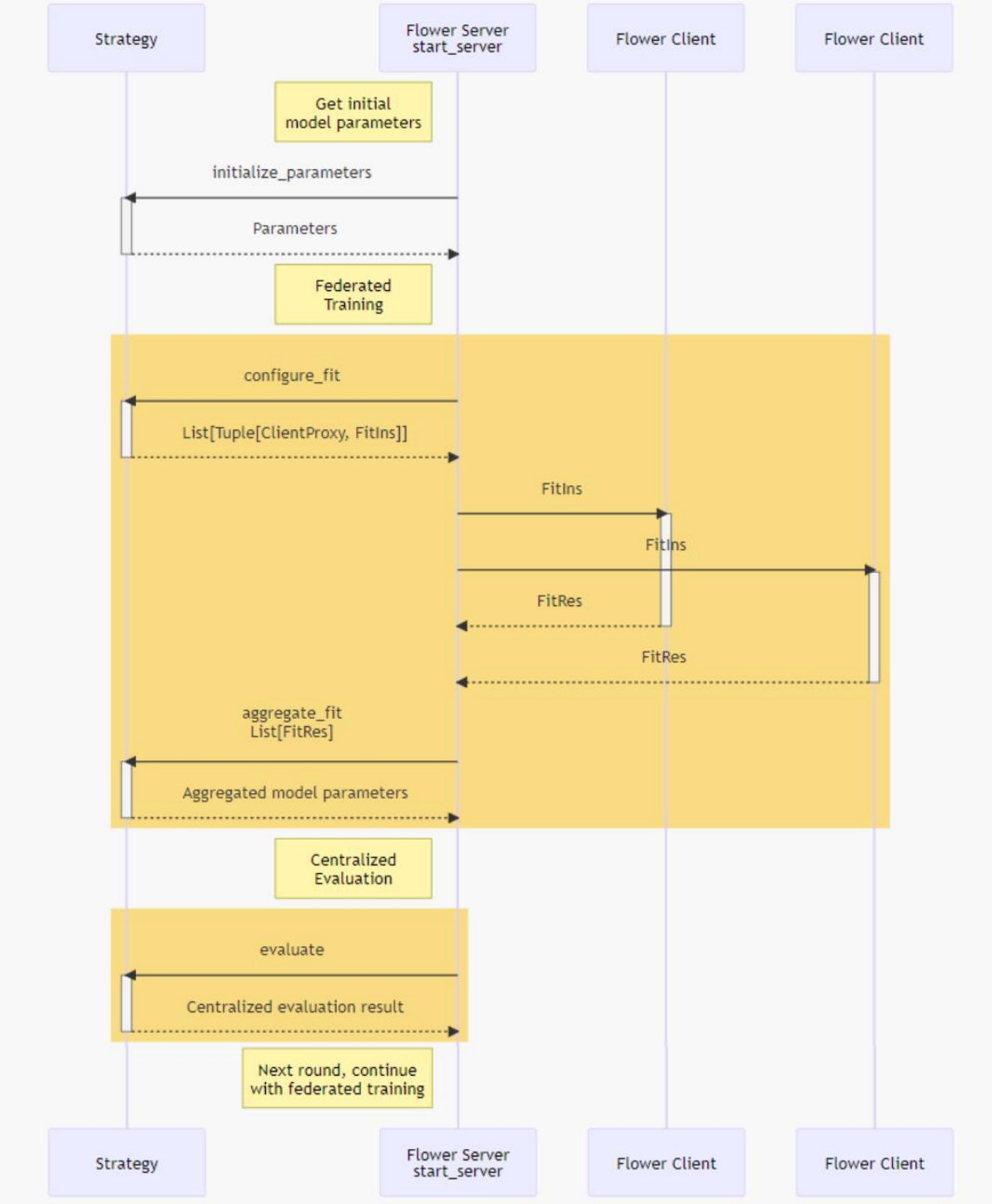


Figure 5.9 Sequence Diagram of Federated Learning Process

5.6. Web-based Pneumonia Detection Using Flask and Swagger

We will investigate how to develop a web-based application for pneumonia detection using Flask, Swagger and Flassger in this subsection. The idea is to develop an easy-to-use website where users may upload chest X-ray photos and get predictions about whether the

image shows signs of pneumonia. For real-time predictions, we will use the stored model we acquired from the federated learning server and incorporate it into the web application.

Python has a simple and adaptable web framework called Flask. By offering a straightforward and basic interface, it helps developers to create web apps fast and efficiently. Developers may easily create server-side logic, manage requests and answers, and define routes with Flask. As a result of its modular and flexible architecture, developers can easily add new capabilities. Flask may be used for both small and large scale projects and is very adaptable.

Swagger is an open-source framework that makes the process of creating, constructing, and documenting APIs simpler. It offers a wide range of tools for creating interactive API documentation, making it simpler for users and developers to comprehend the features of the application. Developers can specify request and response structures, build API endpoints, and even test APIs straight from the documentation using Swagger. By doing this, the API is made sure to be well-documented, simple to use, and clear to other programmers or apps.

Swagger improves the development process by offering an intuitive and interactive API documentation interface when paired with Flask. By integrating Swagger into Flask applications, the Flask-Swagger (Flasgger) extension enables developers to automatically produce API documentation based on specified routes and models. Flasgger makes it simpler for users to comprehend and interact with the application's APIs by allowing them to see and test API endpoints straight from the documentation.

Developers are given the tools they need to build online apps with well-documented APIs by Flask, Flasgger, and Swagger. Developers may accelerate the development process, improve collaboration, and give users a user-friendly and interactive interface to explore and test the application's APIs by utilizing the strength of the Flask web framework and the robustness of Swagger's API documentation capabilities.

Algorithm 9. Web-based Implementation

```
model = load_model(MODEL_PATH)
app = Flask(__name__)
Swagger(app)
@app.route('/')
def welcome():
    return "Welcome All"
@app.route('/predict_file', methods=["POST"])
def predict_note_file():
    """Let's Predict Pneumonia
    This is using docstrings for specifications.
    ---
    parameters:
      - name: file
        in: formData
        type: file
        required: true
    responses:
        200:
            description: The output values
```

```
20:     """
21:     img = Image.open(request.files.get("file"))
22:     img = img.convert('RGB')  # Convert to RGB mode
23:     img = img.resize((180, 180))
24:     x = np.array(img)
25:     x = np.expand_dims(x, axis=0)
26:     x = tf.keras.applications.resnet50.preprocess_input(x)
27:     classes = model.predict(x)
28:     A = np.squeeze(np.asarray(classes))
29:     if A[1] == 1:
30:         return "PNEUMONIA"
31:     else:
32:         return "NORMAL"
```

First, we need to load the saved model from the federated learning server (line 1). Next Flask is utilized to develop the backend for the web application (line 2). In order to handle picture uploads and return predictions based on the loaded model, we will construct routes and endpoints. To document and test our API endpoints, we will incorporate Swagger and Flasgger (line 3). An endpoint for uploading chest X-Ray images within the Flask application is established (line 8). When an image is received, it is first preprocessed before being sent through the loaded model for prediction (line 21-26). The image will be analyzed by the model to see if it suggests the existence of pneumonia and then return the prediction to the webpage (line 29).

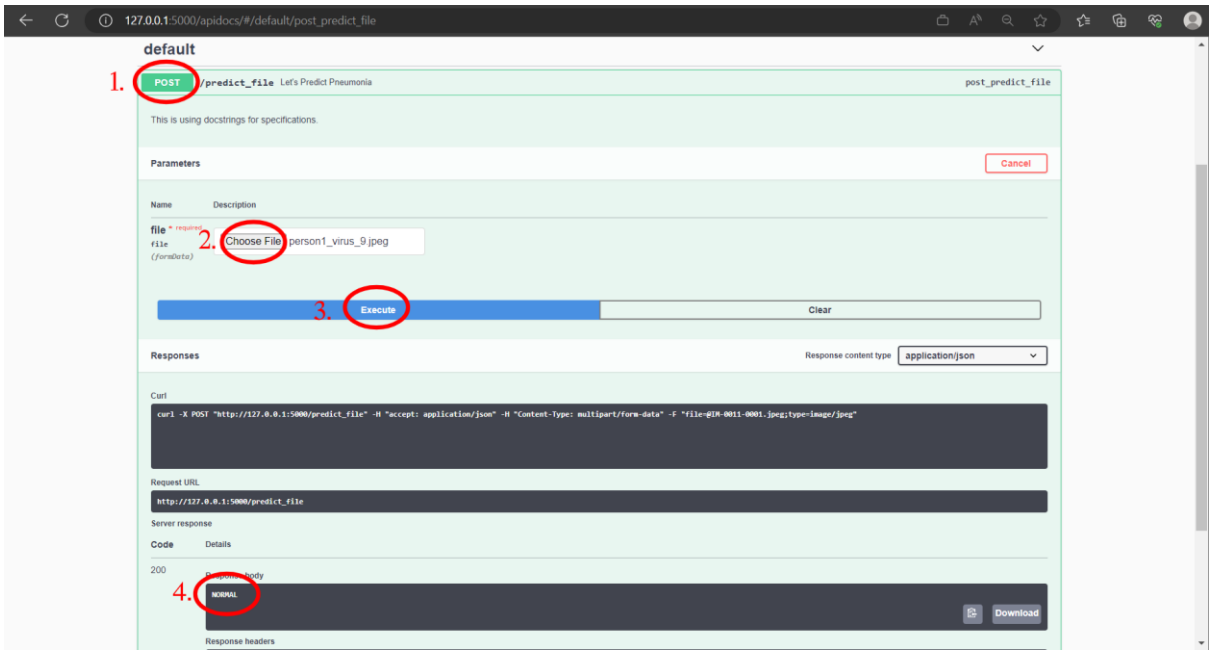


Figure 5.10 Web Application

By copying the link from the console and adding "/apidocs" at the end (in this case, "127.0.0.1:5000/apidocs"), one can view the web application. The next page reached is the one from Figure 5.13, where 1 represents the earlier-built path. A picture must be uploaded at step 2 before pressing the execute (3) button. The prediction then shows up in the Response Body window (4).

5.7. Containerizing the Pneumonia Detection System with Docker

A portable and effective containerization solution is offered by Docker. To ensure consistency and reproducibility across many settings, containers are isolated environments that enclose the application and its dependencies. Because manual setup and dependency management are no longer necessary, it is simpler to deploy the application on a variety of systems with few incompatibilities.

Scalability and portability are made simple using Docker. The project may be put together as a Docker image that has all the required setups and parts. There is no need for laborious setup procedures because this image can be distributed and deployed on any computer that has Docker installed. In addition, Docker supports horizontal scalability, which enables many instances of the program to operate concurrently to meet a rise in client connections or traffic.

A modular and disconnected architecture is encouraged by Docker. The system's clients, servers, and web interface, for example, can all be containerized separately. This modular design makes it simpler to maintain, upgrade, and scale individual components without affecting the system. It also makes it possible for team members working on various project components to collaborate and reuse code more effectively.

Docker offers improved isolation and security. Applications and the dependencies on them are protected from host systems and other containers by running in isolated environments provided by containers. By prohibiting unwanted access and interference with the underlying infrastructure, this improves security.

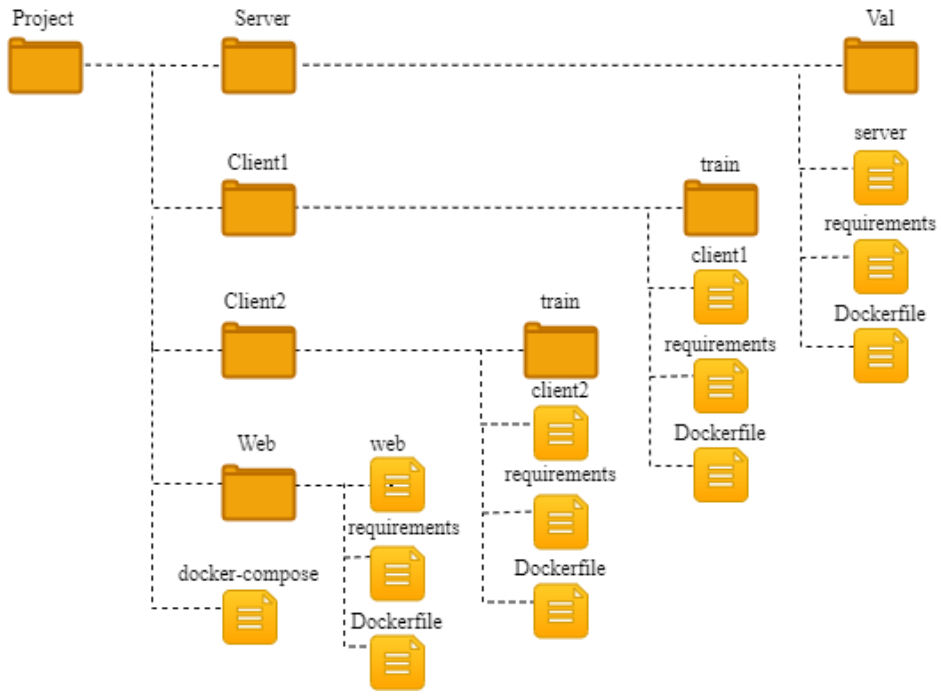


Figure 5.11 Folder Structure for Docker

Docker and Docker Compose can be used to containerize the pneumonia detection system and speed up the deployment procedure. In this method, there are 4 separate Dockerfiles made: one for the server, one for the web interface, and two for the clients. The model weights are stored from the server and loaded into the web interface using an external volume that is shared by the server and web interface. To exchange data and carry out federated learning tasks, the clients will communicate with the server. The multi-container application will be defined and managed using Docker Compose. The structure can be observed in Figure 5.11.

The Dockerfiles for each component must be made first. The client-specific dependencies, code, and configurations will be included in the Dockerfiles for the clients. Like the client Dockerfile, the server Dockerfile will include the parts, settings, and code needed for the server application. The dependencies, code, and configurations needed to operate the web interface and communicate with the server will also be included in the Dockerfile for the web interface. The necessary dependencies are stored in "requirements.txt" for each container.

The multi-container application will then be defined in a Docker Compose file. The services for the clients, server, and web interface will be specified in the Docker Compose file. Each service's build context and Dockerfile will be specified, enabling Docker to create the corresponding images. The volume configuration to share the external volume between the server and web interface will also be included in the Docker Compose file, allowing the loading and saving of model weights.

Algorithm 10. Docker-compose file

```
1:  version: '3'
3:  services:
4:    server:
5:      build: ./server
6:      ports:
7:        - 50051:50051
8:      volumes:
9:        - pneumonia:/app/data
10:   client1:
11:     build: ./client1
12:     environment:
13:       SERVER_ADDRESS: server:50051
14:   client2:
15:     build: ./client2
16:     environment:
17:       SERVER_ADDRESS: server:50051
18:   web:
19:     build: ./web
20:     ports:
21:       - 8080:80
22:     volumes:
23:       - pneumonia:/app/data
24: sysctls:
25:       - net.ipv6.conf.all.disable_ipv6=1
26:
```

The server is built using the “server” directory, it exposes the port 50051 to allow communication with the clients and mounts an external volume called “pneumonia” to “/app/data” for saving and loading the model weights. The clients are built using “./client1” and “./client2” directories. They are configured with the “SERVER_ADDRESS” variable set to “server:50051”, which specifies the address of the server they must connect. The Dockerfile found in the "./web" directory is used to build the "web" service. It mounts the same volume “pneumonia” to "/app/data" for accessing the shared model weights and exposes port 8080 to give access to the web interface. Because the IP generated inside the web docker container is IPv6 it must be disabled (line 25-26).

The deployment and management procedure are made more effective and dependable by containerizing the components of the pneumonia detection system and using Docker Compose. The shared volume enables seamless data exchange, and the separated containers guarantee component encapsulation and remove compatibility problems. This containerized approach encourages modularity, scalability, and portability, making it possible to quickly deploy the pneumonia detection system across many contexts and effectively manage it.

5.8. Discussion of the Results

In-depth examination of the pneumonia detection system was presented in the previous chapters, which covered a number of topics including model choice, evaluation metrics, federated learning implementation, web-based deployment, and containerization using Docker. We will now summarize the main conclusions and offer insights into the relevance and importance of the research in the Discussion of the Results.

Performance metrics for the various model architectures, including VGG16, ResNet50, and InceptionV3, varied. Even though VGG16 had better precision values and less false negatives, ResNet50 won out owing to its overall performance and training duration. ResNet50 demonstrated competitive accuracy, recall, and precision values thanks to its effective use of convolutional layers and appropriate filter sizes. To achieve accurate pneumonia identification, the model's capacity to recognize minute details and extract critical features from chest X-ray pictures was essential.

Table 5.2 Metrics of the different models

Model	Accuracy	Precision	Recall	F1 Score
VGG16	0.89	0.90	0.93	0.91
ResNet50	0.87	0.92	0.86	0.89
InceptionV3	0.88	0.88	0.93	0.91

A distributed and collaborative training approach was created with the use of federated learning and the Flower library. Each client trained locally on its dataset while frequently exchanging model updates with the server in the system, which included a server and numerous clients. The server was able to obtain a global model that included information from all clients thanks to the Federated Averaging (FedAvg) technique, which made it easier to aggregate model weights. The evolution of the accuracy of the aggregated model can be observed in Figure 5.12, with the y-axis being the accuracy and the x-axis the training rounds. This federated learning approach demonstrated the potential for decentralized, privacy-preserving

machine learning, especially in situations where data cannot be centralized owing to regulatory or privacy issues. An in-depth analysis will be done in the next chapter.

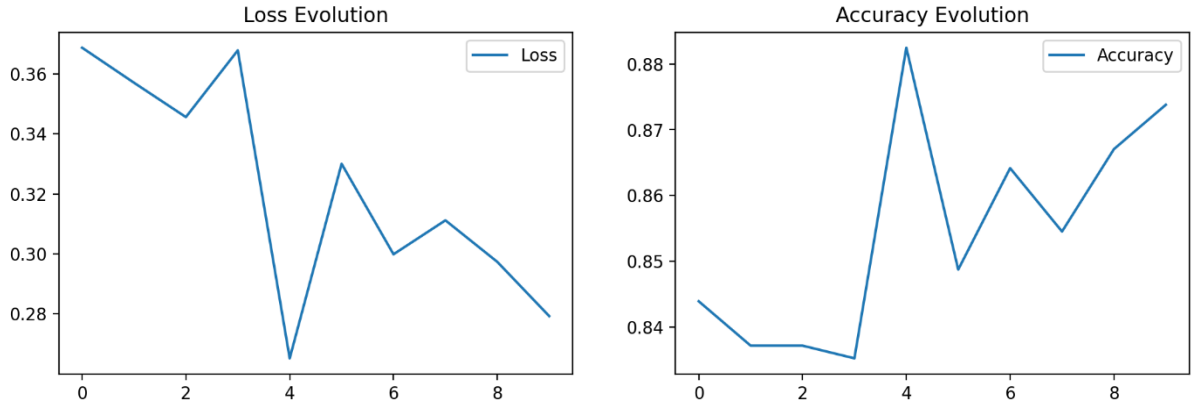


Figure 5.12 Evolution of the Global Accuracy

Users can upload chest X-ray photos and receive real-time forecasts thanks to the web-based deployment of the pneumonia detection system using Flask, Swagger, and Flassger. A user-friendly and interactive web interface was made possible by the combination of Flask and Swagger, and Flassger made it easier to specify and document API endpoints. The web application accomplished accurate and effective pneumonia detection by utilizing the saved model weights from the federated learning server, providing a practical tool for medical practitioners and researchers.

Using Docker to containerize the system improved its portability and scalability. Modular deployment was made possible using independent Docker containers for the server, clients, and web interface. The shared volume for model weight storage-maintained consistency and accessibility, and the Docker Compose file made managing the containers easier. The system's Docker container architecture is shown in Figure 5.13.

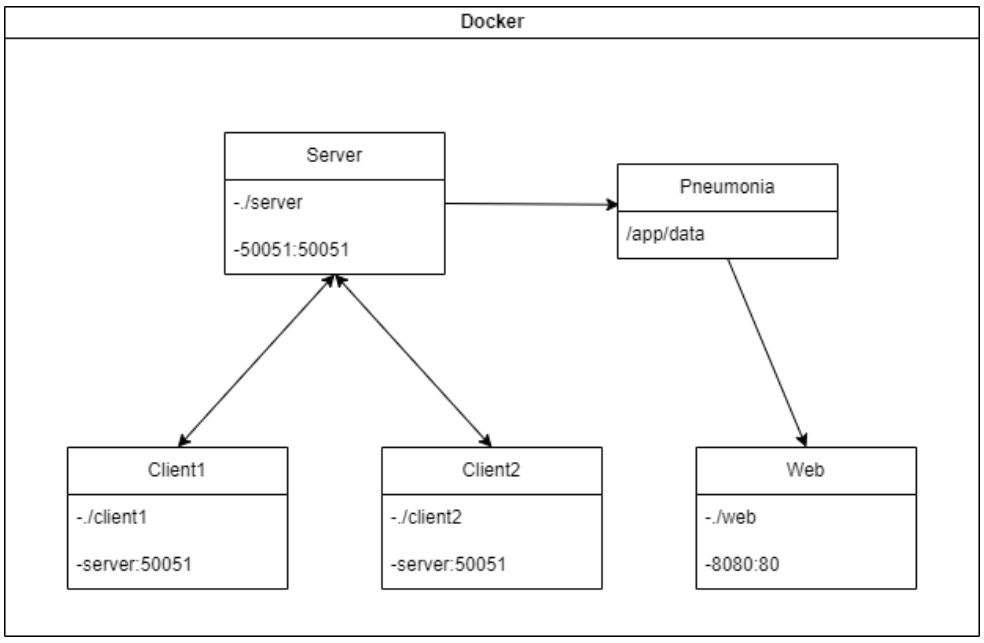


Figure 5.13 Docker Architecture

In conclusion, this project's outcomes show how machine learning models, federated learning, web-based deployment, and containerization are beneficial for detecting pneumonia. ResNet50 was chosen as the model architecture because it offers balanced performance, and federated learning makes it possible for collaborative training to take place while yet protecting data privacy. A user-friendly platform for pneumonia prediction is provided via the web interface, and containerization with Docker improves the system's portability and scalability. Plots, tables, and pictures are used throughout the analysis to support the conclusions and to provide visual representation. Overall, this initiative illustrates how cutting-edge methods could be used to tackle significant healthcare concerns.

Chapter 6. Testing and Validation

The testing and validation findings of the pneumonia detection system employing three distinct models—VGG16, InceptionV3, and ResNet50—will be shown in this chapter. We will examine the confusion matrices and several metrics to assess the performance of each model. We will also go over the federated learning system's training procedure using the selected model (ResNet50), demonstrating the accuracy attained after each round of training and the overall training duration. To show how the web application works, we will upload 10 photographs and compare the predicted values to the actual values.

6.1. Performance Metrics of the Models

We generated different measures, including accuracy, precision, recall, and F1 score, to evaluate the models' performance. To see the classification results visually, the confusion matrix was also created. We can learn more about how well the algorithms can categorize pneumonia patients by examining these data and the confusion matrix.

We acquire a thorough evaluation of the model's performance on the complete dataset by evaluating it on the model resulted from the last round of training, which considers the contributions from both clients. With this strategy, we can better assess the model's performance in identifying pneumonia cases by capturing the collective knowledge of the distributed clients.

The following performance metrics were achieved:

Table 6.1 VGG16 performance metrics

	Precision	**Recall**	**F1-score**	**Support**
Normal	0.89	0.83	0.86	235
Pneumonia	0.90	0.94	0.92	391
Accuracy			0.90	626
Macro Avg	0.90	0.89	0.89	626
Weighted Avg	0.90	0.90	0.90	626

Table 6.2 ResNet50 performance metrics

	Precision	**Recall**	**F1-score**	**Support**
Normal	0.80	0.88	0.84	235
Pneumonia	0.92	0.87	0.89	391
Accuracy			0.87	626
Macro Avg	0.86	0.87	0.86	626
Weighted Avg	0.88	0.87	0.87	626

Table 6.3 InceptionV3 performance metrics

	Precision	**Recall**	**F1-score**	**Support**
Normal	0.89	0.80	0.84	235
Pneumonia	0.89	0.94	0.91	391
Accuracy			0.89	626
Macro Avg	0.89	0.87	0.88	626
Weighted Avg	0.89	0.89	0.89	626

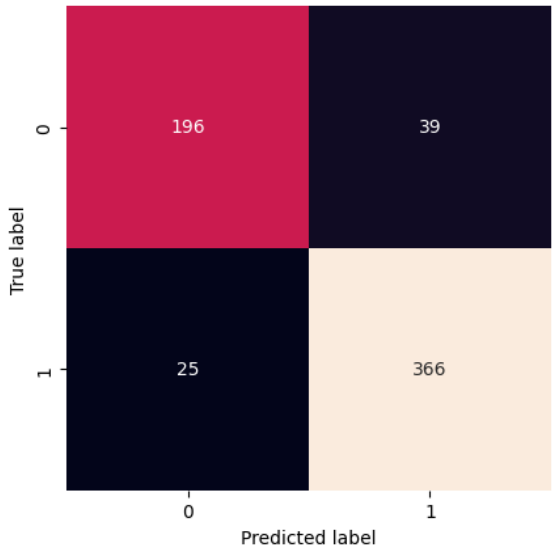


Figure 6.1 VGG16 Confusion Matrix

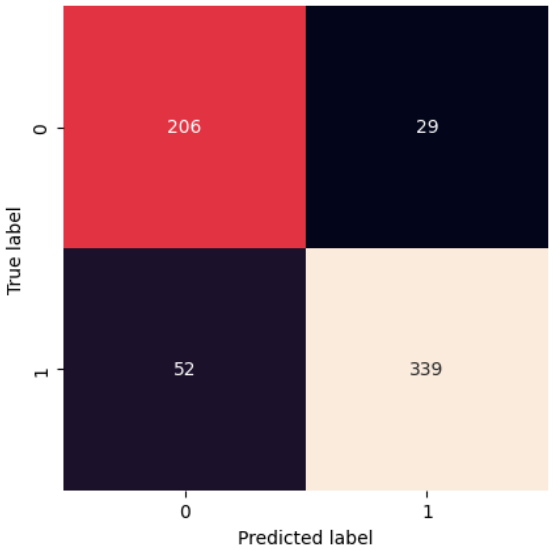


Figure 6.2 ResNet50 Confusion Matrix

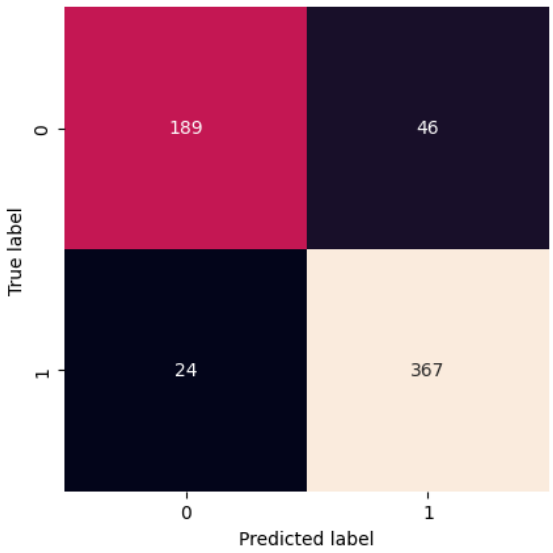


Figure 6.3 InceptionV3 Confusion Matrix

Each model was trained for 10 rounds of federated training; VGG16 required roughly 57 minutes per round, ResNet50 needed 31 minutes per round, and InceptionV3 needed about 18 minutes per round. After training, we used the test dataset to assess the models and generate the classification results.

The classification report for VGG16 showed recall values of 0.89 and 0.90, respectively, with precision values of 0.83 for normal and 0.94 for pneumonia, normal and pneumonia f1-scores were 0.86 and 0.92, respectively, for an overall accuracy of 0.90.

For normal and pneumonia, ResNet50 attained precision scores of 0.80 and 0.92, respectively, with recall values of 0.88 and 0.87. The f1-scores for normal and pneumonia were 0.84 and 0.89, respectively, for a total accuracy of 0.87.

For normal and pneumonia, InceptionV3 produced precision values of 0.89 and 0.89, respectively, with recall values of 0.80 and 0.94. The f1-scores for normal were 0.84 and pneumonia were 0.91, respectively, for a total accuracy of 0.89.

ResNet50 has been chosen as the ideal model for identifying pneumonia. ResNet50 achieved the best accuracy in pneumonia prediction while the other models showed more precision. ResNet50 exhibits a balance between recall, precision, and accuracy, making it a trustworthy option for diagnosing pneumonia. ResNet50 also demonstrates effective training time, which is important for clinical applications. ResNet50 is an effective model for pneumonia identification because of its capacity to capture fine features and robustness in detecting pneumonia patients.

6.2. Transfer Learning with ResNet50

The training of the federated learning system using the ResNet50 model will be covered in this part. We will concentrate on the entire training time as well as the accuracy attained after each training session. Multiple clients can train their local models simultaneously using the federated learning approach, while the server aggregates their weights to produce a global

model. We can boost model performance while maintaining data privacy by using this collaborative training approach.

The training was done using two clients, each with 2096 images. After 10 epochs of training on each client, they send the model to the server where is aggregated and evaluated on a dataset of 1038 images. After the aggregation and evaluation is finished, the server sends the weights of the aggregated model back to the clients and the training is repeated 9 more times.

This simulation was done on an Intel® Core™ i7-6820HQ CPU with 2.70 GHz and 16 GB RAM memory laptop.

The federated server terminal's logs produce useful information on system performance and the training process. These logs keep track of crucial details including the training's progress, round-wise accuracy, loss values, and any potential mistakes or warnings that may have occurred.

```
Found 1038 images belonging to 2 classes.
INFO flwr 2023-06-20 09:52:48,770 | app.py:148 | Starting Flower server, config: ServerConfig(num_rounds=10, round_timeout=None)
INFO flwr 2023-06-20 09:52:48,770 | app.py:168 | Flower ECE: gRPC server running (10 rounds), SSL is disabled
INFO flwr 2023-06-20 09:52:48,770 | server.py:86 | Initializing global parameters
INFO flwr 2023-06-20 09:52:48,770 | server.py:273 | Requesting initial parameters from one random client
INFO flwr 2023-06-20 09:52:51,689 | server.py:277 | Received initial parameters from one random client
INFO flwr 2023-06-20 09:52:51,697 | server.py:88 | Evaluating initial parameters
1038/1038 [==============================] - 79s 74ms/step - loss: 0.7550 - accuracy: 0.2563
INFO flwr 2023-06-20 09:54:11,090 | server.py:91 | initial parameters (loss, other metrics): 0.7549841403961182, {'accuracy': 0.25626203417778015}
INFO flwr 2023-06-20 09:54:11,090 | server.py:101 | FL starting
DEBUG flwr 2023-06-20 09:54:11,091 | server.py:218 | fit_round 1: strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-06-20 10:23:32,191 | server.py:232 | fit_round 1 received 2 results and 0 failures
WARNING flwr 2023-06-20 10:23:32,655 | fedavg.py:243 | No fit_metrics_aggregation_fn provided
WARNING flwr 2023-06-20 10:23:33,104 | fedavg.py:243 | No fit_metrics_aggregation_fn provided
Saving round 1 weights...
1038/1038 [==============================] - 76s 73ms/step - loss: 0.2663 - accuracy: 0.8950
INFO flwr 2023-06-20 10:24:50,438 | server.py:119 | fit progress: (1, 0.26628628373146057, {'accuracy': 0.8949903845787048}, 1839.3232269000146)
DEBUG flwr 2023-06-20 10:24:50,439 | server.py:168 | evaluate_round 1: strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-06-20 10:24:54,196 | server.py:182 | evaluate_round 1 received 2 results and 0 failures
WARNING flwr 2023-06-20 10:24:54,196 | fedavg.py:274 | No evaluate_metrics_aggregation_fn provided
```

Figure 6.3 Server logs for the first round of training

The server logs produced after a training session are shown in Figure 6.3. The dataset and Flower server's startup with 10 training rounds can be seen here. Prior to performing any assessment using client-gathered weights, the global parameters are initialised. After finishing all of these, the training starts with the sampling of 2 clients. The weights obtained after the first round of training are saved once the model has been aggregated, and then there is a second round of evaluation. The "fit_progress" displays the round's loss, accuracy, and duration since the start of training. This is repeated for the next 9 rounds.

```
Saving round 9 weights...
1038/1038 [==============================] - 77s 74ms/step - loss: 0.2754 - accuracy: 0.8767
INFO flwr 2023-06-20 14:30:27,831 | server.py:119 | fit progress: (9, 0.2753771543502807 6, {'accuracy': 0.8766859173774719}, 16576.32619339996)
DEBUG flwr 2023-06-20 14:30:27,831 | server.py:168 | evaluate_round 9: strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-06-20 14:30:29,980 | server.py:182 | evaluate_round 9 received 2 results and 0 failures
DEBUG flwr 2023-06-20 14:30:29,981 | server.py:218 | fit_round 10: strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-06-20 14:59:28,733 | server.py:232 | fit_round 10 received 2 results and 0 failures
Saving round 10 weights...
1038/1038 [==============================] - 78s 75ms/step - loss: 0.2801 - accuracy: 0.8719
INFO flwr 2023-06-20 15:00:48,034 | server.py:119 | fit progress: (10, 0.2801254689693451, {'accuracy': 0.8718689680099487}, 18396.5069944)
DEBUG flwr 2023-06-20 15:00:48,035 | server.py:168 | evaluate_round 10: strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-06-20 15:00:50,101 | server.py:182 | evaluate_round 10 received 2 results and 0 failures
INFO flwr 2023-06-20 15:00:50,101 | server.py:147 | FL finished in 18398.583633399976
INFO flwr 2023-06-20 15:00:50,134 | app.py:218 | app_fit: losses_distributed [(1, 0.21691011637449265), (2, 0.2480821013450622 6), (3, 0.2384013235569000 2),
(4, 0.2753637954592705), (5, 0.2426446825265884 4), (6, 0.2347791492938995 4), (7, 0.2502527311444282 5), (8, 0.2358737140893936 2), (9, 0.2133914232254028 3), (10, 0.255982168018817
INFO flwr 2023-06-20 15:00:50,135 | app.py:219 | app_fit: metrics_distributed_fit {}
INFO flwr 2023-06-20 15:00:50,135 | app.py:220 | app_fit: metrics_distributed {}
INFO flwr 2023-06-20 15:00:50,136 | app.py:221 | app_fit: losses_centralized [(0, 0.7549841403961182), (1, 0.2662862837314605 7), (2, 0.2764509022235870 4),
(3, 0.2790813148021698), (4, 0.2971042990684509), (5, 0.2704315483570099), (6, 0.2638399004936218), (7, 0.3008021116256714), (8, 0.2839798927307129),
(9, 0.2753771543502807 6), (10, 0.2801254689693451)]
INFO flwr 2023-06-20 15:00:50,136 | app.py:222 | app_fit: metrics_centralized {'accuracy': [(0, 0.2562620341777801 5), (1, 0.8949903845787048), (2, 0.884393036365509),
(3, 0.8728323578834534), (4, 0.8689787983894348), (5, 0.8882466554641724), (6, 0.8805394768714905), (7, 0.8641618490219116), (8, 0.8786126971244812), (9, 0.8766859173774719),
(10, 0.8718689680099487)]}
```

Figure 6.4 Server logs for the last rounds

The same procedure is followed for the last rounds (Figure 6.4). The overall training time is displayed following the saving of the weights generated from the previous cycle.

Following that, each of the metrics produced (loss, accuracy) is saved along with the training cycle.

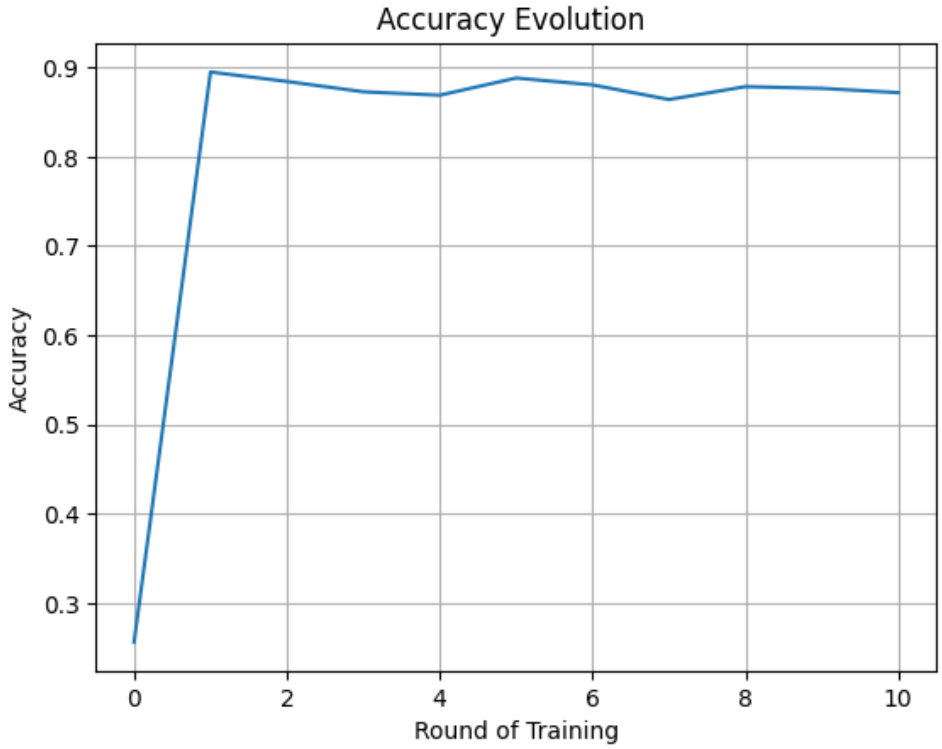


Figure 6.4 Accuracy evolution during the rounds of training

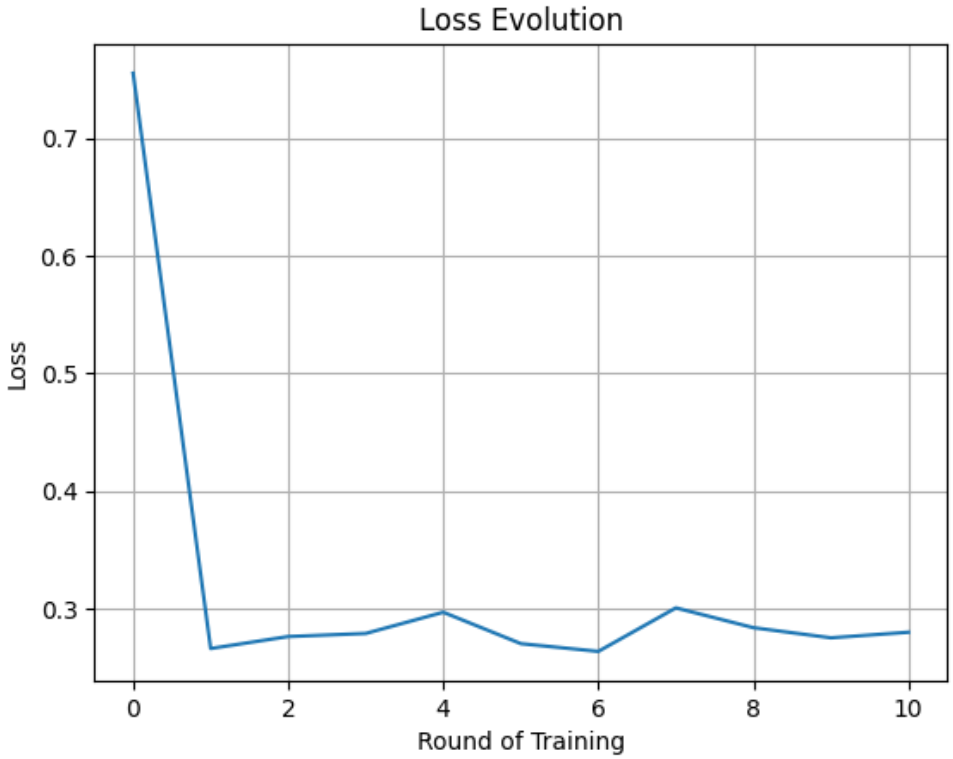


Figure 6.5 Loss evolution during the rounds of training

The accuracy and loss graph can be used to gain important insights into how well the model performed during training. We can track the model's capacity to provide accurate

predictions across a series of rounds by looking at the accuracy curve. By analyzing the graph, it is observed that the max accuracy was obtained in the first round of training (0.89) and the minimum loss is obtained in the sixth round of training (0.26). Also, it is observed that the model converges at round 7 by reaching a plateau after which accuracy struggles to improve and remains approximately at the same value.

Table 6.4 Duration per Round of Training

Round	1	2	3	4	5	6	7	8	9	10
Time (s)	1839	1827	1816	1826	1828	2272	1820	1858	1850	1820

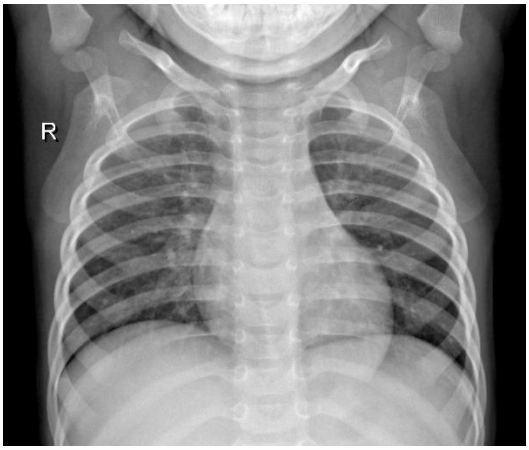
The length of each cycle is a key factor in determining how effective the training program is. We can learn more about computing needs and resource usage by measuring the amount of time it takes for each round. The total duration of the ten rounds of training is 306 minutes, with an average of 31 minutes and 15 seconds. Planning, resource allocation, and system scalability assessments can all benefit from this data. Additionally, knowing the overall training duration offers a benchmark for contrasting various training strategies or improving the training pipeline. We may learn a lot about the effectiveness of the training process in terms of both time and performance by considering the round length, overall training time, and average duration.

6.3. Web Application Testing

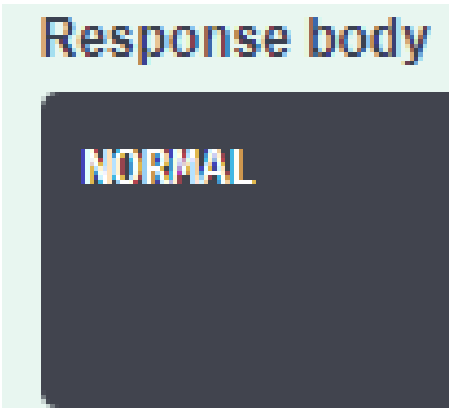
We carried out testing by uploading 10 chest X-ray images (5 normal and 5 with pneumonia) and contrasting the predictions with the actual values to assess the web application's functionality. This testing phase offers information about the system's functionality in the actual world and its precision in identifying pneumonia cases. We will review the predicted outcomes, evaluate the accuracy, and go over any issues that came up during the testing process.

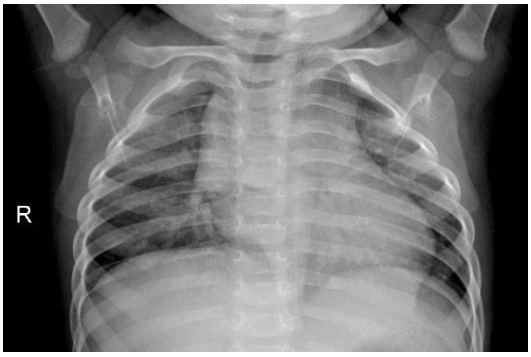
The chest X-Ray image tested:

The Prediction:

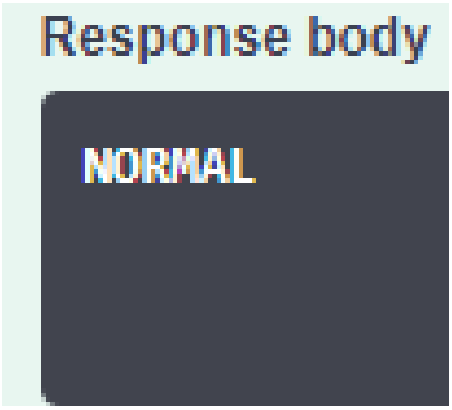


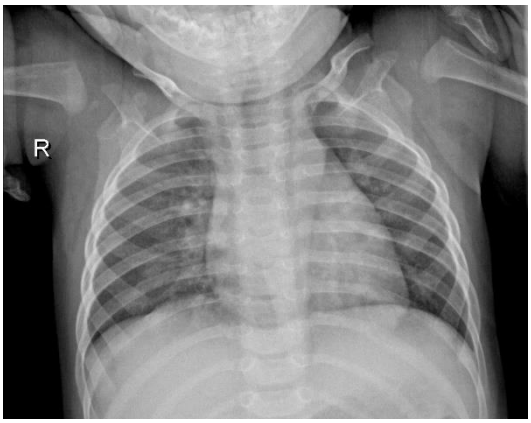
Actual label: Normal Chest X-Ray 1



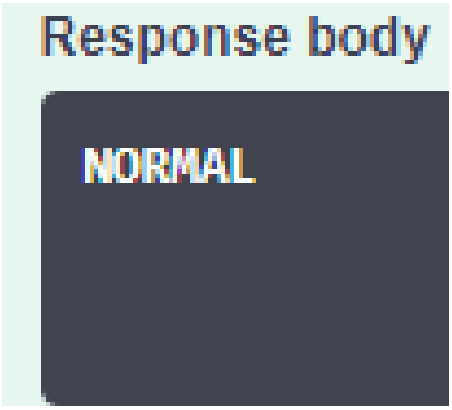


Actual label: Normal Chest X-Ray 2

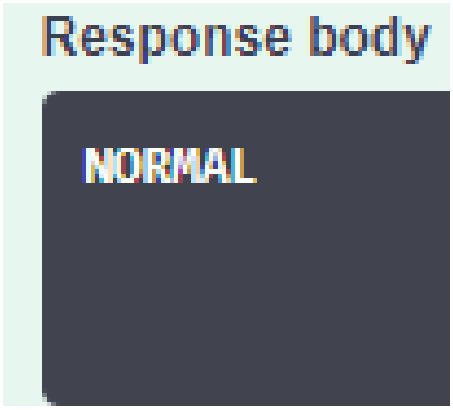




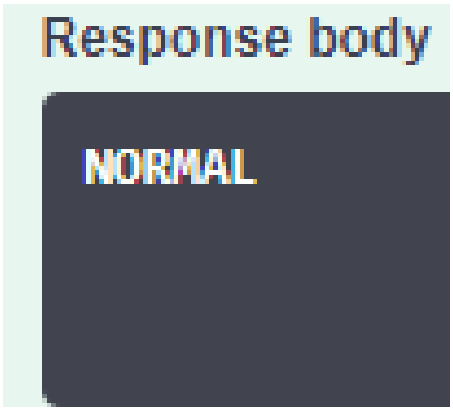
Actual label: Normal Chest X-Ray 3

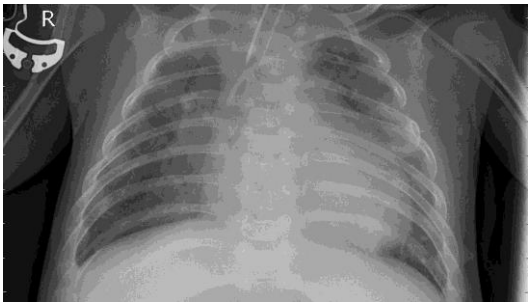


Actual label: Normal Chest X-Ray 4

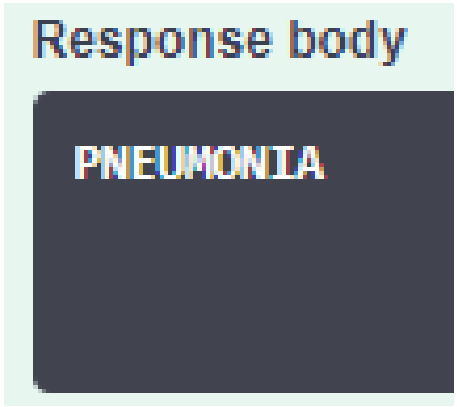


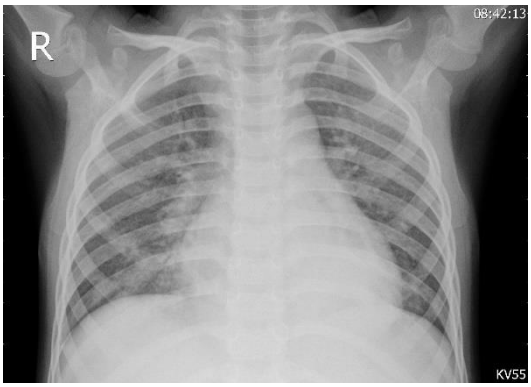
Actual label: Normal Chest X-Ray 5



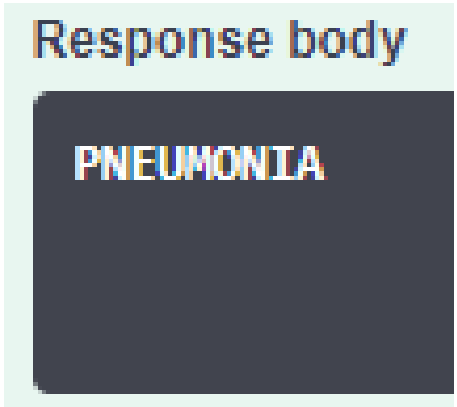


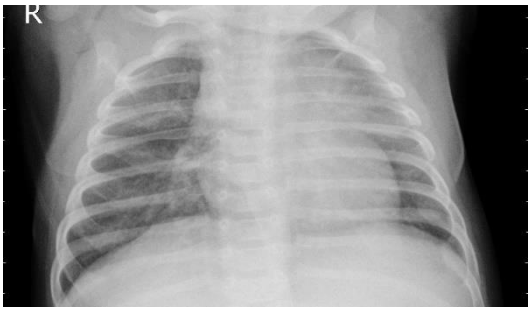
Actual label: Pneumonia Chest X-Ray 1



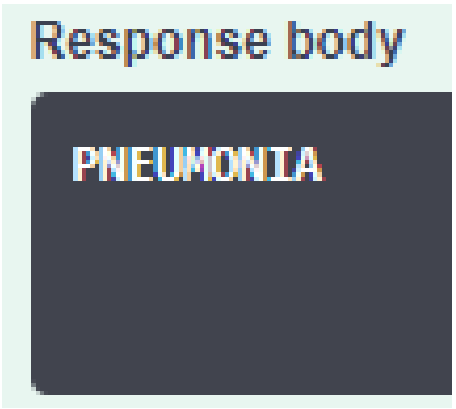


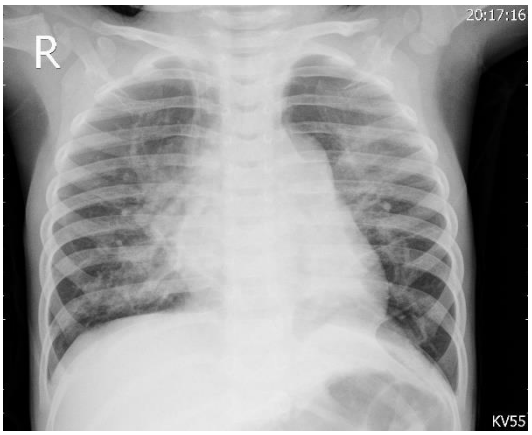
Actual label: Pneumonia Chest X-Ray 2



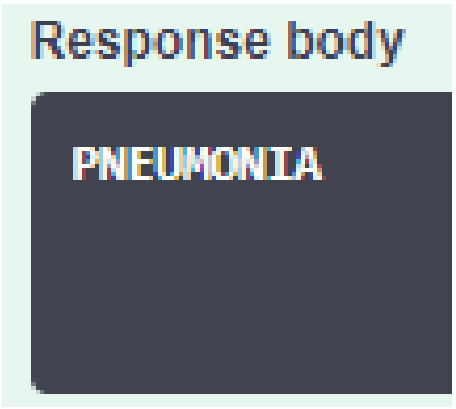


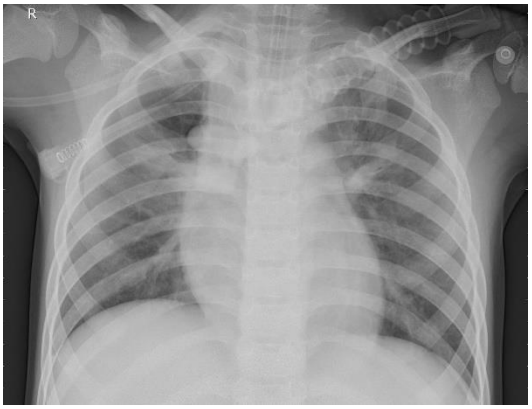
Actual label: Pneumonia Chest X-Ray 3



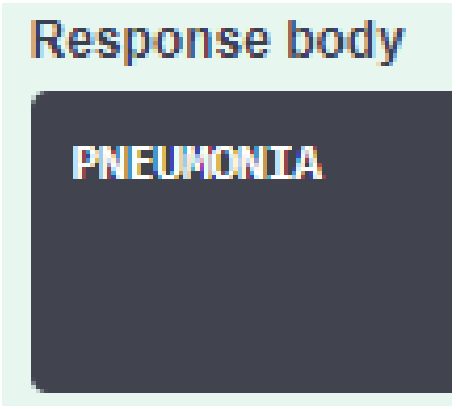


Actual label: Pneumonia Chest X-Ray 4





Actual label: Pneumonia Chest X-Ray 5



All ten of the test images that were loaded for classification were successfully categorized by the online application. Both cases of healthy lungs and the ones with pneumonia were classified correctly. Also, for testing purposes, a “txt” file was also uploaded. When uploading non images files (e.g. jpg, jpeg, png) the “Choose File” changes its color to red and the “Execute” button cannot be pressed.

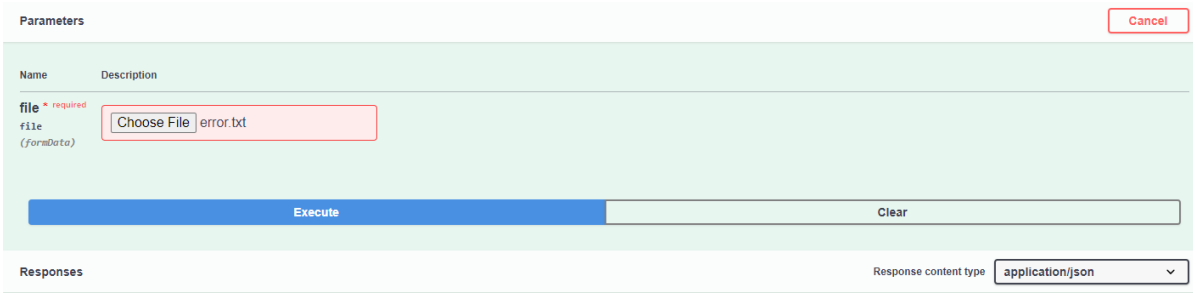


Figure 6.6 Choose file error

Overall, evaluating the effectiveness and dependability of the pneumonia detection system heavily relies on the testing and validation phase. We can confidently assess the system's efficacy and make defensible choices for future improvements by analyzing the confusion matrix, monitoring the accuracy during federated learning training, and performing real-world testing on the web application.

Chapter 7. User's Manual

This manual gives you a step-by-step walkthrough on how to operate the system, upload chest X-ray images, and get predictions on whether pneumonia is present. Docker is used to containerize the system, making it portable and smooth across many operating systems. This user's guide will help you set up and utilize the pneumonia detection system successfully, regardless matter whether you are a healthcare professional, researcher, or an individual curious about exploring AI's potential in medical diagnostics.

First the entire project must be copied to a folder on the local machine. Next make sure that Docker is installed on the computer and that the system is compatible with Docker. Docker is compatible with most operating systems, but for Windows it requires virtualization support to be enabled in the system BIOS and Hyper-V to be installed. For windows it is recommended to use Docker Desktop. Next open a terminal or command prompt and navigate to the project directory. After reaching the project folder, run the following command to start the Docker containers: "docker-compose up". This will build the necessary images and start the containers for server, clients, and web application. Next run in the terminal "docker logs web" to be able to see the output of the container and copy the URL generated there to the web browser. You will be greeted with the homepage of pneumonia detection website.

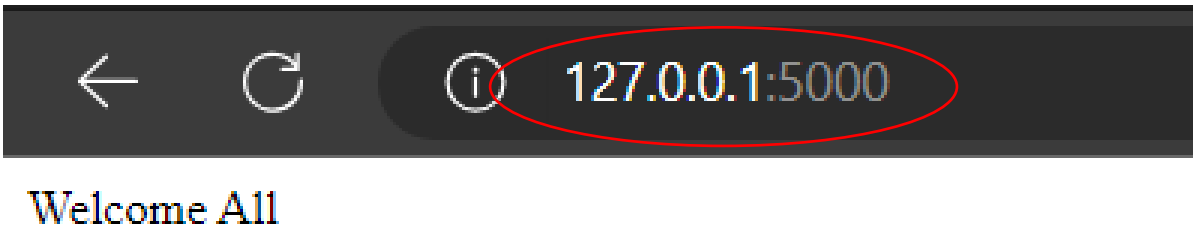


Figure 7.1 Homepage

Next add at the end of the URL (the marked location in Figure 7.1) "/apidocs". This will open the page in Figure 7.2.

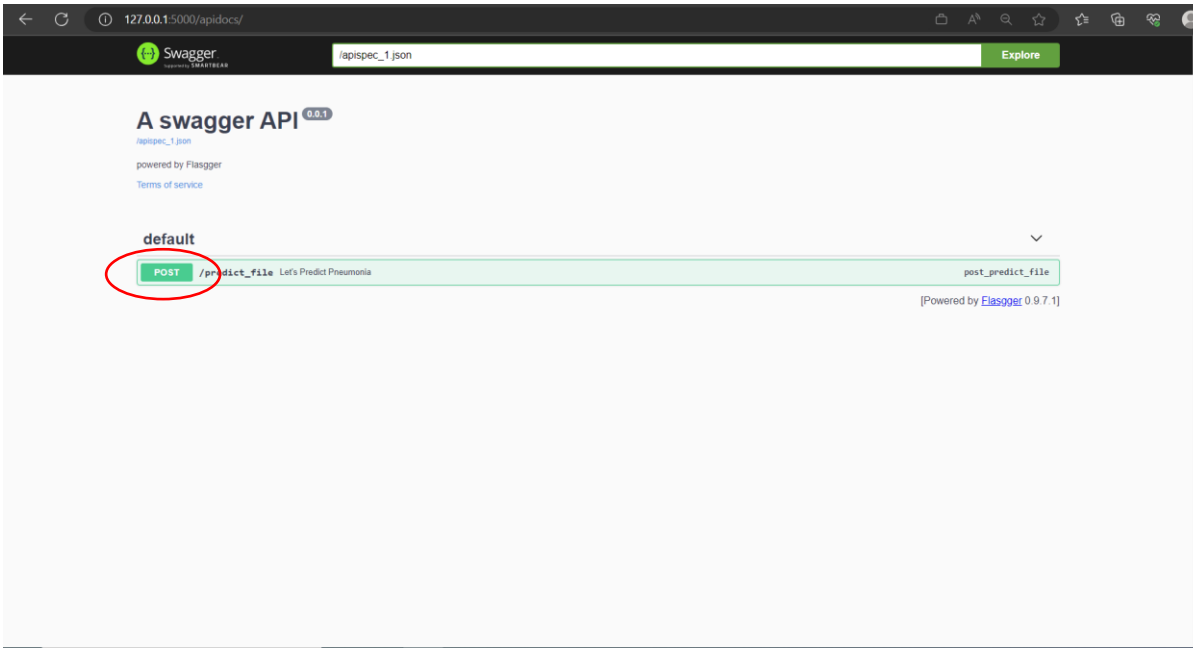


Figure 7.2 Pneumonia Detection Webpage

Then press "POST" or anywhere on the light blue region. After that the windows will expand downwards and will look like in Figure 7.3.

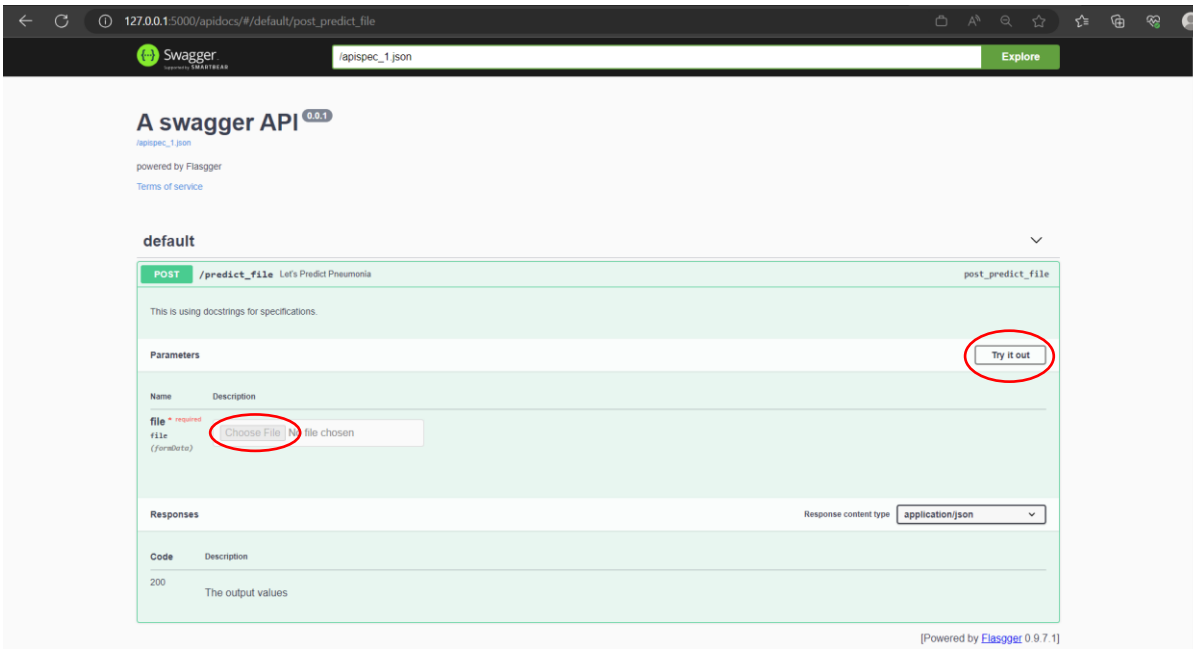


Figure 7.3 Insert image form

Then press "Try it out", after which the button "Choose File" will no longer be greyed out. Press that button and a window will appear where an image of Chest X-Ray will be selected and uploaded.

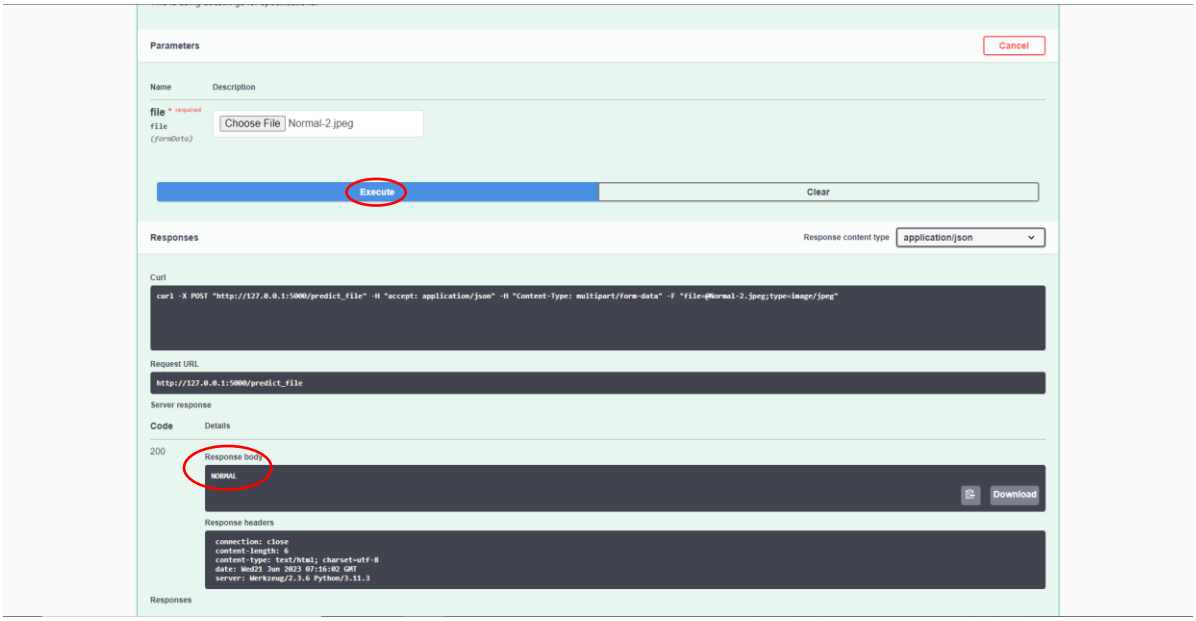


Figure 7.4 Execution and Response

After the image is uploaded a blue button with the text "Execute" will appear, press it and after about a second the prediction will appear in the "Response Body" field.

To upload another file, press clear and then repeat the steps from "Choose File" onwards.

If you wish to change the hyperparameters for the training, navigate to the specific directory. Open the Python files (e.g. client1.py), and at the beginning of the file the change the values of the variables with "hyper" at the beginning of them. After which save the changes and rebuild the Docker containers using the "docker-compose up –build" in the terminal.

By following these instructions, you can use the web interface for the pneumonia detection system, upload photos for prediction, and adjust the model's performance's hyperparameters to meet your needs.

Chapter 8. Conclusions

8.1. Summary of the Findings

The main goal of this study was to create a reliable and accurate pneumonia detection system. To do this, machine learning methods and web-based user interfaces were used. Multiple models were carefully evaluated for the study, and the ResNet50 model ultimately proved to be the best option. It was a prime choice for pneumonia detection due to its outstanding accuracy and capacity to identify fine details in chest X-ray pictures.

Federated learning was used during the training process, and it produced excellent outcomes. Federated learning enabled group model training while protecting the confidentiality and security of sensitive patient information. The ResNet50 model demonstrated the convergence required for precise pneumonia identification through repeated training rounds. This convergence demonstrated that the model had correctly identified and mastered the key patterns and traits connected to pneumonia cases.

A user-friendly web-based interface was created to increase the system's usability. Users were able to upload their chest X-ray photos and receive instantaneous predictions for the detection of pneumonia using this user-friendly interface. The web interface supported speedy and accurate diagnosis, assisting in effective healthcare decision-making by offering a smooth and engaging experience.

The ability of machine learning to enhance pneumonia detection was demonstrated by the combination of the ResNet50 model, federated learning, and the user-friendly online interface. The study proved the viability of the selected model, the advantages of federated learning for collaborative training, and the significance of user-centric design in healthcare applications. This system's successful integration opens the door to improvements in pneumonia detection technologies as well as the possible use of related approaches for treating other medical diseases.

8.2. Contributions of the Study

Additionally, by emphasizing the advantages of using deep learning models, this study advances pneumonia detection systems. Evaluation and comparison of various architectures, including VGG16 and InceptionV3, reveals the benefits and drawbacks of each in terms of spotting pneumonia cases. The decision to use ResNet50 as the preferred model highlights its greater functionality and capacity to capture complex elements in chest X-ray pictures, resulting in more precise diagnoses.

This study's use of federated learning makes a substantial addition to the discipline of medical image analysis. Federated learning tackles privacy issues while utilizing the combined expertise and data resources of several healthcare providers. It does this by enabling collaborative training across numerous institutions without revealing sensitive patient data. This strategy not only increases the model's reliability and accuracy but also encourages a wider adoption of machine learning methods in healthcare settings.

The creation of an accessible web interface improves the usability and accessibility of the system for detecting pneumonia. The web interface gives customers the convenience of uploading chest X-ray images and providing real-time forecasts, giving healthcare providers an important tool for rapid and accurate decision-making. The user-friendly interface and intuitive design guarantee ease of use and encourage seamless integration into current clinical workflows.

Additionally, by showcasing the potential of deep learning and web-based interfaces in enhancing diagnostic accuracy and effectiveness, this study contributes to the broader area of medical picture analysis. Future study and development in the field of computer-aided diagnosis for pneumonia and other medical disorders will be built on the findings and approaches described here.

Although this study makes important discoveries and advances, it is important to recognize its limitations and pinpoint possible topics for further research.

8.3. Limitations and Future Work

Even though this study marked advancements in the identification of pneumonia, it is crucial to recognize its shortcomings and point out areas in need of further development. The generalizability of the established system is a restriction to consider. The model's performance was assessed using a particular dataset, and its efficacy may change when used with various populations or imaging technologies. To guarantee the system's durability and reliability across various clinical situations, future studies could try to test the system using larger and more diversified datasets from multiple sources.

The web-based interface can be improved further to increase functionality even though it offers a user-friendly platform for image submission and prediction. The system's overall performance might be enhanced, for instance, by adding an image preprocessing module to handle image resizing, normalization, and noise reduction. Additionally, implementing a feedback mechanism that enables users to offer feedback on anticipated outcomes might aid in enhancing the model's accuracy and fine-tuning over time.

The investigation of various aggregation techniques in the federated learning framework is another potential area for future development. In this investigation, the model updates from several clients were combined using a straightforward average aggregation approach. However, there are several different aggregation methods that can be researched, including weighted averaging, importance-based aggregation, and secure aggregation. By allocating varying weights to client updates based on their dependability or relevance, these techniques may enhance the performance of the model. The federated learning process could also be improved by investigating adaptive aggregation systems, which dynamically modify the aggregate process based on client performance or data properties. Researchers can improve the performance of the federated learning system and obtain greater convergence, accuracy, and robustness by examining and contrasting alternative aggregating techniques.

Future research should concentrate on resolving these issues to further progress pneumonia detection systems, considering the constraints and potential areas for improvement. This may entail partnering with healthcare organizations to gain access to bigger and more varied datasets as well as utilizing cutting-edge technologies like deep learning architectures

and sophisticated preprocessing methods. Additionally, investigating the integration of additional imaging modalities or alternative diagnostic tools can result in pneumonia diagnoses that are more thorough and precise.

In conclusion, this study makes significant advancements and contributions to the field of pneumonia identification. It not only reaches significant accuracy and usability milestones, but it also identifies areas that require additional study and development. Researchers and practitioners can continue to progress pneumonia detection systems and ultimately enhance patient care and outcomes in medical imaging and diagnosis by addressing the observed limits and looking into the suggested future directions.

Bibliography

[1] Rani S. Gereige, Pablo Marcelo Laufer; *Pneumonia*. *Pediatr Rev* October 2013; 34 (10): 438–456. https://doi.org/10.1542/pir.34-10-438

[2] E. Ayan and H. M. Ünver, "Diagnosis of Pneumonia from Chest X-Ray Images Using Deep Learning," *2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)*, Istanbul, Turkey, 2019, pp. 1-5, doi: 10.1109/EBBT.2019.8741582.

[3] E. Çallı, E. Sogancioglu, B. van Ginneken, K. G. van Leeuwen, and K. Murphy, "Deep learning for chest X-ray analysis: A survey," *Medical Image Analysis*, vol. 72, p. 102125, Aug. 2021, doi: 10.1016/j.media.2021.102125.

[4] T. Li, A. K. Sahu, A. Talwalkar and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," in IEEE Signal Processing Magazine, vol. 37, no. 3, pp. 50-60, May 2020, doi: 10.1109/MSP.2020.2975749.

[5] J. Kaur, "Federated Learning Applications and Its Working | 2022," *XenonStack*, Jun. 27, 2022. https://www.xenonstack.com/blog/federated-learning-applications

[6] Torres, A., Cilloniz, C., Niederman, M.S. *et al.* Pneumonia. *Nat Rev Dis Primers* 7, 25 (2021). https://doi.org/10.1038/s41572-021-00259-0

[7] "Pneumonia - Symptoms and causes - Mayo Clinic," Mayo Clinic, Jun. 13, 2020. https://www.mayoclinic.org/diseases-conditions/pneumonia/symptoms-causes/syc-20354204

[8] *Tatiana Gabruseva, Dmytro Poplavskiy, Alexandr Kalinin*; Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2020, pp. 350-351

[9] Esteva, A., Kuprel, B., Novoa, R. *et al.* Dermatologist-level classification of skin cancer with deep neural networks. *Nature* 542, 115–118 (2017). https://doi.org/10.1038/nature21056

[10] Gulshan V, Peng L, Coram M, Stumpe MC, Wu D, Narayanaswamy A, Venugopalan S, Widner K, Madams T, Cuadros J, Kim R, Raman R, Nelson PC, Mega JL, Webster DR. Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. JAMA. 2016 Dec 13;316(22):2402-2410. doi: 10.1001/jama.2016.17216. PMID: 27898976.

[11] Beaulieu-Jones BK, Orzechowski P, Moore JH. Mapping Patient Trajectories using Longitudinal Extraction and Deep Learning in the MIMIC-III Critical Care Database. Pac Symp Biocomput. 2018;23:123-132. PMID: 29218875.

[12] C. Réda, E. Kaufmann, and A. Delahaye-Duriez, "Machine learning applications in drug development," Computational and Structural Biotechnology Journal, vol. 18, pp. 241–252, 2020, doi: 10.1016/j.csbj.2019.12.006.

[13] Murphy, R. An active role for machine learning in drug development. *Nat Chem Biol* 7, 327–330 (2011). https://doi.org/10.1038/nchembio.576

[14] Bacchi, S., Tan, Y., Oakden-Rayner, L., Jannes, J., Kleinig, T. and Koblar, S. (2022), Machine learning in the prediction of medical inpatient length of stay. Intern Med J, 52: 176-185. https://doi.org/10.1111/imj.14962

[15] I. Y. Chen, E. Pierson, S. Rose, S. Joshi, K. Ferryman, and M. Ghassemi, "Ethical Machine Learning in Healthcare," *Annual Review of Biomedical Data Science*, vol. 4, no. 1, pp. 123–144, Jul. 2021, doi: 10.1146/annurev-biodatasci-092820-114757.

[16] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, vol. 149, p. 106854, Nov. 2020, doi: 10.1016/j.cie.2020.106854.

[17] M. Ruby, "How ChatGPT Works: The Model Behind The Bot - Towards Data Science," *Medium*, May 07, 2023. https://towardsdatascience.com/how-chatgpt-works-the-models-behind-the-bot-1ce5fca96286

[18] *Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. ACM Trans. Intell. Syst. Technol. 10, 2, Article 12 (March 2019), 19 pages. https://doi.org/10.1145/3298981*

[19] Rieke, N., Hancox, J., Li, W. *et al.* The future of digital health with federated learning. *npj Digit. Med.* 3, 119 (2020). https://doi.org/10.1038/s41746-020-00323-1

[20] S. H. Khan and M. G. R. Alam, "A Federated Learning Approach to Pneumonia Detection," *2021 International Conference on Engineering and Emerging Technologies (ICEET)*, Istanbul, Turkey, 2021, pp. 1-6, doi: 10.1109/ICEET53442.2021.9659591.

[21] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. Ch. Paschalidis, and W. Shi, "Federated learning of predictive models from federated Electronic Health Records," *International Journal of Medical Informatics*, vol. 112, pp. 59–67, Apr. 2018, doi: 10.1016/j.ijmedinf.2018.01.007.

[22] Makkar, A., Santosh, K. SecureFed: federated learning empowered medical imaging technique to analyze lung abnormalities in chest X-rays. *Int. J. Mach. Learn. & Cyber.* 14, 2659–2670 (2023). https://doi.org/10.1007/s13042-023-01789-7

[23] He, Chaoyang & Annavaram, Murali & Avestimehr, Salman. (2020). FedNAS: Federated Deep Learning via Neural Architecture Search. https://doi.org/10.48550/arXiv.2004.08546

[24] S. Mascarenhas and M. Agarwal, "A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification," *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, Bengaluru, India, 2021, pp. 96-99, doi: 10.1109/CENTCON52345.2021.9687944.

[25] A. Deshpande, V. V. Estrela, and P. Patavardhan, "The DCT-CNN-ResNet50 architecture to classify brain tumors with super-resolution, convolutional neural network, and the ResNet50," *Neuroscience Informatics*, vol. 1, no. 4, p. 100013, Dec. 2021, doi: 10.1016/j.neuri.2021.100013.

[26] Jignesh Chowdary, G., Punn, N.S., Sonbhadra, S.K., Agarwal, S. (2020). Face Mask Detection Using Transfer Learning of InceptionV3. In: Bellatreche, L., Goyal, V., Fujita, H., Mondal, A., Reddy, P.K. (eds) Big Data Analytics. BDA 2020. Lecture Notes in Computer Science(), vol 12581. Springer, Cham. https://doi.org/10.1007/978-3-030-66665-1_6

[27] T. Li, A. K. Sahu, A. Talwalkar and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," in *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50-60, May 2020, doi: 10.1109/MSP.2020.2975749.

[28] Rieke, N., Hancox, J., Li, W. *et al.* The future of digital health with federated learning. *npj Digit. Med.* 3, 119 (2020). https://doi.org/10.1038/s41746-020-00323-1

[29] Li, Wenqi & Milletari, Fausto & Xu, Daguang & Rieke, Nicola & Hancox, Jonny & Zhu, Wentao & Baust, Maximilian & Cheng, Yan & Ourselin, Sébastien & Cardoso, Manuel Jorge & Feng, Andrew. (2019). Privacy-Preserving Federated Brain Tumour Segmentation. 10.1007/978-3-030-32692-0_16.

[30] D. S. Kermany *et al.*, "Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning," *Cell*, vol. 172, no. 5, pp. 1122-1131.e9, Feb. 2018, doi: 10.1016/j.cell.2018.02.010.