AI-Driven Multi-Class Classification of X-ray Images for Disease Detection

By

Cheang Jia Zhi

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION SYSTEMS (HONOURS)
DIGITAL ECONOMY TECHNOLOGY
Faculty of Information and Communication Technology
(Kampar Campus)

JUNE 2025

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Dr. Abdulkarim Kanaan Jebna, for providing me with this valuable opportunity to undertake this project. His patient guidance, insightful feedback, and consistent support throughout this journey have been instrumental in the completion of this work. I am truly grateful for his expertise and mentorship that have helped shape my understanding of deep learning in image classification.

I would also like to extend my heartfelt thanks to my family for their unwavering support, encouragement, and love throughout my academic journey. Their belief in my capabilities and constant motivation have been a source of strength during challenging times.

ABSTRACT

There are critical challenges in medical imaging diagnostics, including human error in radiological analysis, slow diagnostic processes, and radiologist shortages. This project developed an AI-driven multi-class classification system for chest X-ray disease detection. The research employed a DenseNet121 architecture with transfer learning to classify 13,482 chest X-ray images into five categories: COVID-19, pneumonia, tuberculosis, lung opacity, and normal cases. The methodology implemented a two-stage progressive unfreezing strategy, comprehensive data preprocessing with augmentation techniques, and class weight balancing to address dataset imbalances. Training utilized TensorFlow and Keras frameworks with GPU acceleration, incorporating early stopping and learning rate reduction callbacks for optimization. Gradient-weighted Class Activation Mapping (Grad-CAM) was integrated for AI interpretability, and a comprehensive Streamlit dashboard was developed featuring real-time processing capabilities. The DenseNet121 model achieved exceptional performance with 94.52% validation accuracy, 94.7% test accuracy, 95% precision and recall, and 0.99 AUC score across all disease categories. The system successfully demonstrated clinical-grade interpretability through visual attention mapping and deployed as a functional web application with hospital finder and weather health monitoring features. This research establishes a foundation for AI-assisted medical diagnosis, potentially improving healthcare accessibility and diagnostic reliability while maintaining transparency in AI decision-making processes for clinical deployment.

Area of Study: Artificial Intelligence, Medical Imaging

Keywords: Deep Learning, Chest X-ray Classification, Transfer Learning, Lung Disease Detection, Computer-Aided Diagnosis

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CNNs	Convolutional Neural Networks
CXR	Chest X-ray
COVID-19	Coronavirus Disease 2019
GPU	Graphics Processing Unit
Grad-CAM	Gradient-weighted Class Activation Mapping
COPD	Chronic Obstructive Pulmonary Disease
CRISP-DM	Cross-Industry Standard Process for Data Mining
ROC	Receiver Operating Characteristic
CT	Computed Tomography
MRI	Magnetic Resonance Imaging

CHAPTER 1

Introduction

This chapter will start with background information, providing an overview of AI in medical imaging, historical development of AI in image classification and the relevance to Chest X-ray (CXR) analysis. Then follow by the problem statement and motivation, highlighting the need for automated disease detection. Next, I will discuss the project's objectives, followed by its scope and direction. The contributions of this work to healthcare will also be emphasized, along with a brief overview of the report's organization.

1.1 Background Information

Artificial Intelligence (AI) has transformed medical imaging by automating the analysis of diagnostic images like X-rays, Computed Tomography (CT), and Magnetic Resonance Imaging (MRI). Through leveraging cutting-edge techniques in machine learning and deep learning, AI can recognize complex patterns, improving diagnostic accuracy and aiding clinicians in detecting various conditions across fields like radiology, pathology, and cardiology [1]. This automation streamlines workflows, enables faster diagnoses, support timely patient care and improve patient outcomes [1]. AI's capabilities in medical imaging have been significantly enhanced by the development of Convolutional Neural Networks (CNNs), which excel at image classification.

The rise of Convolutional Neural Networks (CNNs) in the early 2010s marked a significant leap in image classification capabilities. The field surged in 2012 when a pioneering CNN architecture called AlexNet was introduced by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. AlexNet achieved breakthrough performance in image classification by leveraging deeper architectures and GPUs [2]. Subsequent models, such as VGGNet in 2014 and ResNet in 2016, introduced deeper networks and residual connections, further reducing error rates. More advanced architectures like DenseNet and MobileNet were also introduced in recent years [3]. Transfer learning later enabled CNNs to excel in specialized domains like medical imaging by adapting pre-trained models to smaller datasets, thereby advancing automated image analysis.

Lung diseases pose a significant threat to human lives, lung disease like Pneumonia accounted for 15% of global deaths in children under five years old in 2017 [4]. So explicitly, timely detection of lung conditions is crucial for accelerating recovery and improving long-term survival chances [5]. A chest X-ray (CXR), a medical imaging technique that uses low-dose radiation to visualize the lungs and chest cavity, serves as a vital tool for diagnosing diseases such as pneumonia, tuberculosis, lung cancer, and, more recently, COVID-19. Compared to Computed Tomography (CT) scans, CXRs are more affordable, safer, and involve lower radiation exposure, making them widely accessible [6]. The emergence of COVID-19 in 2019, which caused severe respiratory damage with symptoms appearing up to 14 days post-infection and required 2–4 hours for diagnosis, further underscored the need for rapid and effective diagnostic tools like CXRs to enable timely intervention [7].

This paper discusses how to use deep learning algorithms to classify Chest X-ray images (CXR) into distinct lung conditions. I get the dataset from Kaggle which is public available datasets to distinguish five classes of Chest X-ray (COVID-19, Pneumonia, Tuberculosis, Lung Opacity, and Normal Cases). I use a pre-trained architectures - DenseNet121, to get the robust classification performance.

1.2 Problem Statement and Motivation

The application of AI in medical imaging, particularly for chest X-ray analysis, faces significant challenges in achieving reliable disease detection. **Human error in manual analysis** remains a critical issue, as subjective interpretations by radiologists often lead to **misdiagnosis,** causing delays in patient care and unnecessary interventions [8], [9], [10], [11]. For instance, in the case of Chronic Obstructive Pulmonary Disease (COPD), a progressive lung condition characterized by airflow limitation, misdiagnosis is a prevalent problem in primary care settings. From Figure 1.2 (a), we can see that misdiagnosis remains a significant challenge in respiratory disease detection, where 43% of COPD patients in primary care were either incorrectly diagnosed or incorrectly graded [12]. Apparently, addressing human error in manual analysis is vital to enhance diagnostic reliability and improve patient outcomes through more consistent and objective evaluations.

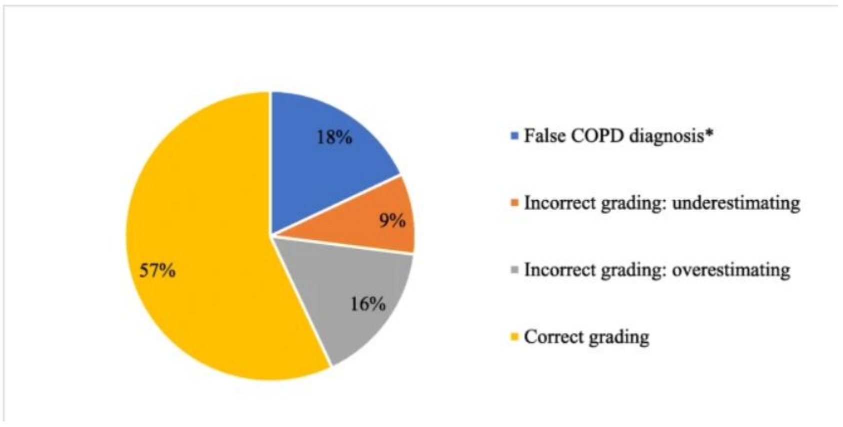


Figure 1.2 (a) Accuracy of diagnosis for COPD . Adapted from [12]

Moreover, the **slow diagnostic processes** hinder rapid and accurate identification of diseases like pneumonia and COVID-19, which is critical in urgent healthcare scenarios [8], [9], [10], [11], [13]. Developing advanced AI models can assist radiologists and enable early diagnosis in underserved regions, improving access to timely healthcare. Additionally, The **scarcity of trained radiologists**, particularly in rural and low-resource areas, often leads to delayed or missed diagnoses, leaving many patients without timely care [11], [15], [22]. From Figure 1.2 (b) below, we can see that the radiologist shortage in the U.S. is a critical issue. The demand for radiologists is increasing due to the aging U.S. population, while the supply remains stagnant [14]. This shortage underscores the urgent need for automated AI-assisted systems that can assist radiologists and enable early diagnosis in underserved regions, ensuring equitable access to healthcare.

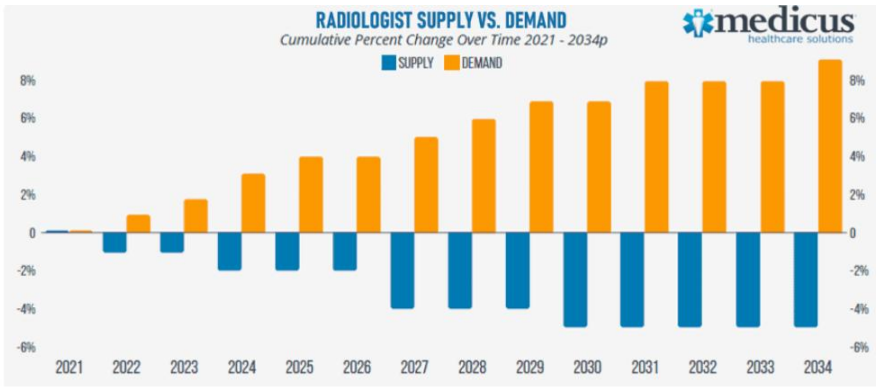


Figure 1.2 (b) Issue of Radiologist Shortage in U.S . Adapted from [14]

Besides, AI models for X-ray disease detection face significant limitations that impede their clinical utility. There is a high risk of **model overfitting on small datasets** due to

limited training data, as highlighted in studies [19], [20]. This risk restricts their ability to generalize to new cases, limiting their practical effectiveness in diverse clinical settings. Common approaches to address overfitting include expanding training datasets through augmentation techniques, applying regularization methods, implementing early stopping during training, and using pre-trained models. Furthermore, current models **often detect only a single disease**, making them ineffective for multi-class classification tasks where multiple pathologies may coexist in a single X-ray [16], [21]. Solving this limitation in disease detection scope is critical to enable comprehensive diagnosis, allowing for more accurate treatment plans and reducing the burden on healthcare systems through efficient, multi-disease detection. My proposed approach addresses this challenge by development robust multi-class classifier for multiple lung disease detection.

1.3 Project Objectives

The aim of the project is to propose a system that addresses the critical need for improved diagnostic tools in the medical field, particularly in radiology, by implementing an AI-driven system for the multi-class classification of X-ray images. The objectives of the project are as follows:

Objective 1: To Develop a Deep Learning Model for Multi-Class Chest X-ray Classification of COVID-19, Pneumonia, Tuberculosis, Lung Opacity, and Normal Cases, Achieving >90% Accuracy.

Objective 2: To Implement Data Augmentation Techniques and Model Optimization Techniques for Improving Model Robustness and Generalization in Medical Chest X-ray Image Classification

Objective 3: To Build a Comprehensive Medical Dashboard System with Real-Time Processing for Chest X-rays.

Objective 4: To Implement AI Interpretability and Explainability Features for Medical Decision Support .

1.4 Project Scope and Direction

This project aims to develop an automated multi-class classification of chest X-ray images to detect lung diseases, specifically focusing on COVID-19, pneumonia, tuberculosis, lung opacity, and normal cases. By implementing a sophisticated AI-driven diagnostic tool, the system provides a web interface where users can upload X-ray images for rapid, accurate analysis. The primary objective is to assist radiologists, reduce diagnostic errors, and enable timely medical interventions, particularly in regions with limited medical expertise.

The solution leverages transfer learning with the DenseNet121 model, emphasizing both accuracy and computational efficiency. By utilizing a progressive unfreezing training strategy, the system mitigates overfitting and enhances diagnostic precision. The implementation focuses on comprehensive performance evaluation using standard metrics like accuracy, precision, recall, and AUC scores. Unique features such as Grad-CAM visualization provide transparency in the AI's decision-making process, building trust and offering clinically relevant insights. The project demonstrates the potential of AI to support medical diagnostics by delivering a reliable, interpretable tool for multi-class lung disease detection, with a particular emphasis on handling class imbalance and maintaining high diagnostic performance across all disease categories.

1.5 Contributions

This project presents an AI-driven multi-class classification system for chest X-ray (CXR) images, offering a transformative solution for lung disease diagnosis across five critical categories: COVID-19, pneumonia, tuberculosis, lung opacity, and normal conditions. The unique contribution lies in implementing a sophisticated transfer learning approach using the DenseNet121 model, with a progressive unfreezing strategy that addresses key challenges in medical image classification.

The system's key innovations include comprehensive attention quality verification through Grad-CAM visualization, demonstrating the model's ability to focus on clinically relevant anatomical regions. By developing a robust methodology that handles class imbalance and provides transparent AI decision-making, the project bridges critical gaps in diagnostic reliability. The implementation goes beyond traditional accuracy metrics, incorporating advanced interpretability features that build trust among healthcare professionals. The dashboard's integrated approach—combining AI diagnosis, hospital finder, and environmental health alerts—creates a holistic healthcare support tool that is particularly valuable in resource-limited settings.

The project's broader impact extends to addressing significant healthcare challenges, including radiologist shortages and diagnostic uncertainties. By providing a fast, accurate, and interpretable diagnostic support system, the research demonstrates the potential of AI to enhance medical diagnostics, improve patient outcomes, and create more accessible healthcare solutions. The scalable and adaptable approach sets a new standard for AI-driven medical imaging, showcasing how advanced machine learning techniques can be translated into practical, life-saving technologies.

1.6 Report Organization

This report is structured into seven chapters detailing the development of an AI-driven system for multi-class chest X-ray disease detection. Chapter 1 introduces the project, outlining the problem of lung disease diagnosis, motivation, objectives, and scope. Chapter 2 reviews existing literature on AI-based chest X-ray analysis, critically analysing prior works and examining current commercial tools to justify the proposed solution. Chapter 3 presents the system methodology using CRISP-DM framework, including design specifications, implementation challenges, and project timeline. Chapter 4 describes the system design through activity diagrams, use case scenarios, and wireframe interfaces. Chapter 5 details the system implementation, covering hardware and software setup, model training configuration using DenseNet121, and dashboard deployment with operational features. Chapter 6 provides comprehensive system evaluation through performance metrics, confusion matrices, ROC analysis, Grad-CAM interpretability assessment, and objective validation. Finally, Chapter 7 concludes with key findings, contributions to healthcare AI, and recommendations for future development.

CHAPTER 2

Literature Reviews

This chapter presents a comprehensive literature review of recent studies and existing tools focused on AI-driven multi-class classification of chest X-ray (CXR) images for disease detection. The review under section 2.1 aims to analyze and categorize existing research based on the methodologies and architecture employed, highlighting some key elements like objectives, datasets, preprocessing techniques, modeling approaches, evaluation metrics, and results. Through reviewing these papers, we can gain insights into the strengths and limitations of existing approaches, setting the foundation for developing more effective and practical solutions for AI-driven lung disease classification.

2.1 Approaches for Multi-Class Chest X-ray Classification

The following sections explore diverse methodological approaches found in scientific literature for categorizing chest X-ray (CXR) images to identify lung diseases. By systematically organizing these research strategies, the analysis reveals the nuanced strengths, innovative techniques, and inherent challenges characterizing current multi-class medical image classification efforts.

2.1.1 Custom-Designed/ Hybrid CNN Based Approaches

This subsection examines studies that implement custom-designed, or hybrid CNN architectures specifically tailored for multi-class chest X-ray classification tasks. T. Ahad et al. [13] proposed a modified DenseNet201 combined with a custom CNN architecture. The paper addresses the need for early detection of lung disease and time-consuming with traditional testing methods. Their model utilized a custom dataset from Kaggle comprising 17,902 CXR images across multiple classes (COVID-19, pulmonary fibrosis, tuberculosis, and normal cases). When compared with other models like ResNet50 and VGG16, their proposed approach achieved an impressive 95.43% testing accuracy. In paper [15], G.H. Daganaw and M.E. Moutlaq proposed a lightweight CNN architecture which is shown in Figure 2.1.1 (a) below. It compared the performance against pre-trained models like DenseNet201 and EfficientNetB0. They also utilized Score-CAM to help for better visual representation and interpreting the decision-making process. The proposed CNN achieved 97.5% accuracy with a precision of 97.4%, sensitivity of 98.6%, and an F1-score of 97.4%.

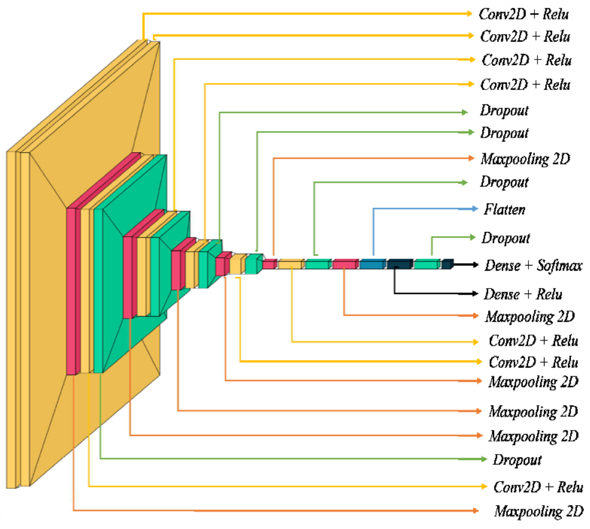


Figure 2.1.1 (a) Proposed CNN architecture . Adapted from [15]

In another paper, Alshmrani, G. M. M. et al [18] proposed VGG19 + CNN (three CNN blocks for feature extraction, followed by fully connected layers for classification). The researchers successfully classified a total of 80,000 CXR images (Covid-19, Lung Opacity, pneumonia, lung cancer, TB, normal) and achieved robust performance with 96.48% accuracy and 97.56% precision. In [19], Abhishek Agnihori and Narendra Kohli introduced three proposed architectures: Novel CNN, hybrid CNN-LSTM, and hybrid CNN-RF. Although CNN-RF model outperformed and achieved an accuracy of 94.66%. The primary weakness in this paper is the limited dataset size which is 1,512 images, this may lead to overfitting and hinder generalization across diverse cases. In paper [23], M. Chetoui et al. proposed fine-tuned EfficientNet-B5 with Swish activation function (DeepCCXR-Bin) achieved outstanding results with an AUC of 0.985 and average sensitivity and specificity of 0.94 and 0.98 respectively.

2.1.2 Transfer Learning with Pre-trained Models Based Approaches

This subsection focuses on studies that utilize transfer learning with pre-trained models, emphasizing the fine-tuning of existing architectures for lung disease classification from CXR images. Rasha S. Gargeges [11] proposed using multiple pre-trained models (ResNet152, DenseNet201, ResNeXt101, SqueezeNet1, VGG19, and AlexNet) to diagnose diseases automatically and efficiently. The paper aimed to address the limited availability of expert radiologists and time-consuming diagnostic processes. Using a novel large multiclass dataset created from several publicly available sources (COVID-19, lung opacity, pneumonia, tuberculosis, normal), the researchers unified image sizes to 224×224 pixels, normalized to ImageNet dataset parameters, and 70:30 ratio split the data into training and validation sets. Researcher trained all the models with 50 epochs and 8 batch sizes for fair comparison. And used performance metrics like Accuracy, FBeta score, Matthews Correlation Coefficient (MCC), and Hamming Loss to evaluate the performance of models. At last, ResNet152 model demonstrated the best performance with 95.643% accuracy, FBeta of 0.968. In another paper [16], Albahlh et al. implemented three pre-trained models which are DenseNet121, InceptionResNetV2, and ResNet152V2. The paper aimed to identify multiple chest-related diseases simultaneously. Tested on 112,120 frontal-view CXR images labeled with 14 diseases. The approach addressed previous models' limitations in detecting only one disease and ineffectiveness due to data insufficiency and imbalanced dataset. In data preprocessing step, images were reshaped to (150,150,3), normalized, 80:20 split into train and test set, augmented, and one-hot encoded for labels. As result, InceptionResNetV2 achieved the highest ROC-AUC score of 0.801.

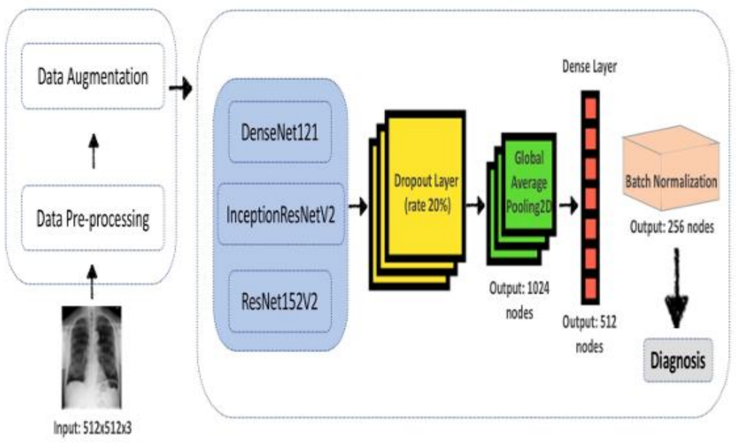


Figure 2.1.2 (a) Overview of the Proposed Work Framework. Adapted from [16]

2.1.3 Ensemble Learning Based Approaches

This subsection explores the studies that employ ensemble learning to combine multiple models, aiming to enhance the accuracy and robustness of lung disease classification from CXR images. K. Vij et al. [9] utilized an ensemble of ResNet and VGG16 on 85,000 CXR images achieved 97.87% precision, 97.62% accuracy. The paper mentioned the issues of mistake prone of manual diagnosis and slow diagnosis process. M. K. Sagar et al. [10] employed an ensemble of Inception-V3 and VGG16 on Kaggle datasets (ChestXRay-8, Pneumonia), resized to 224x224x3, implemented scaling and data augmentation. Weaknesses involve resource constraints limiting model inclusion, a small test set (16 images), and batch size challenges during training. The approach achieved 88.14% accuracy with an emphasis on automation and speed.

Furthermore, as shown in Figure 2.1.3 (a) below, R. Arora et al. [17] proposed a stochastic ensemble of CNNs (DenseNet and GoogleNet) for feature extraction, processing these features through a variational autoencoder and combining predictions from traditional ML classifiers (SVM, Random Forest, XGBoost) with logistic regression as a meta-learner. The model was evaluated on two datasets: Dataset D1 containing 5,840 images (1,575 Normal, 4,265 Pneumonia) and Dataset D2 with 954 images (308 COVID-19, 323 Normal, 323 Pneumonia). By implementing this sophisticated ensemble architecture, they achieved impressive performance metrics with 91.7% accuracy, demonstrating the effectiveness of their hybrid approach in multi-class chest X-ray classification.

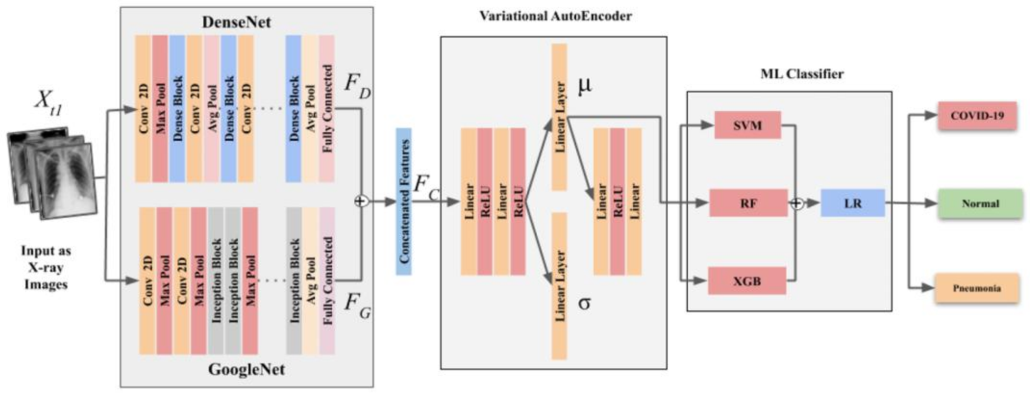


Figure 2.1.3 (a) Proposed Framework . Adapted from [17]

2.1.4 Segmentation-Classification Based Approaches

As shown in Figure 2.1.4 (a), N. Sharma et al. [21] proposed a segmentation-based deep learning model with embedded explainable AI for COVID-19 detection. Their approach combines UNet or UNet+ for lung segmentation with eight deep classifiers (VGG16, VGG19, Xception, InceptionV3, Densenet201, NASNetMobile, Resnet50, and MobileNet) for classification. The methodology addressed existing AI-based detection systems that lack lung segmentation or are limited to binary classification. Using 704 CXR images for segmentation and 12,926 images from three datasets for classification. The researchers selected the best-performing model combination, UNet for segmentation and Xception for classification. UNet + Xception combination achieved an outstanding classification accuracy of 97.45% with an AUC of 0.998. They also achieved an error rate of 2% which passed the regulatory requirements (error rate <5%) for clinical settings.

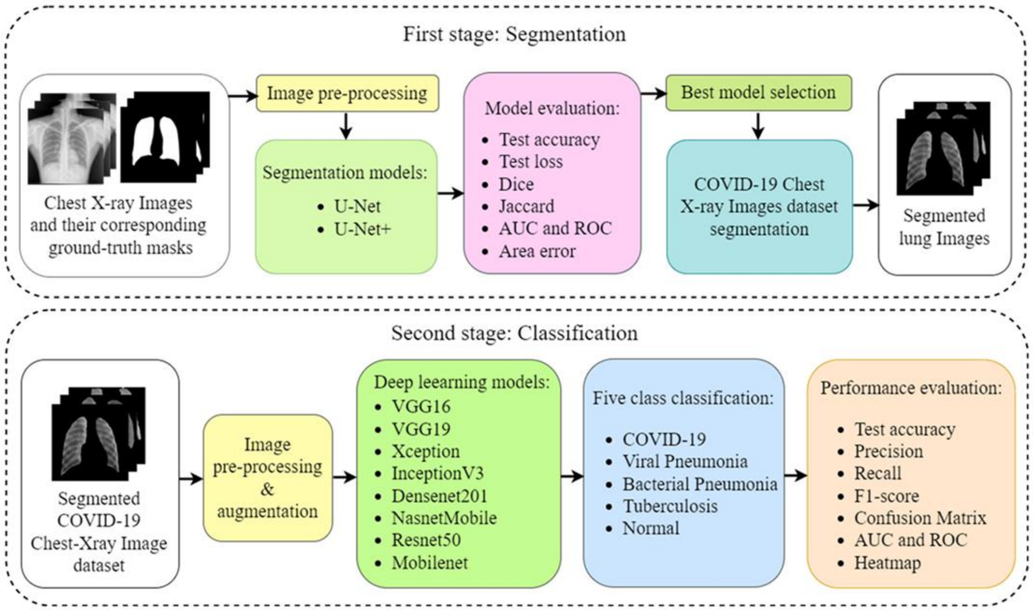


Figure 2.1.4 (a) Overall Schematic Diagram For The Proposed Method . Adapted from [21]

In [22], Rahman et al. developed a reliable tuberculosis detection system using chest X-rays with deep learning segmentation and visualization. Their approach features original and modified U-Net for lung segmentation as shown in Figure 2.1.4 (b), coupled with nine pre-trained CNNs which are ResNet18, ResNet50, ResNet101, ChexNet, InceptionV3, VGG19, DenseNet201, SqueezeNet, and MobileNetV2 for classification. The paper mentioned the issue about the shortage of trained radiologists in resource-limited areas and frequent misclassification of TB cases. Testing on 7,000 images (3,500 TB and 3,500 normal), their ChexNet achieved 96.47% accuracy without segmentation, while DenseNet201 reached 98.6% accuracy with segmentation, demonstrating that segmentation significantly improved classification performance.

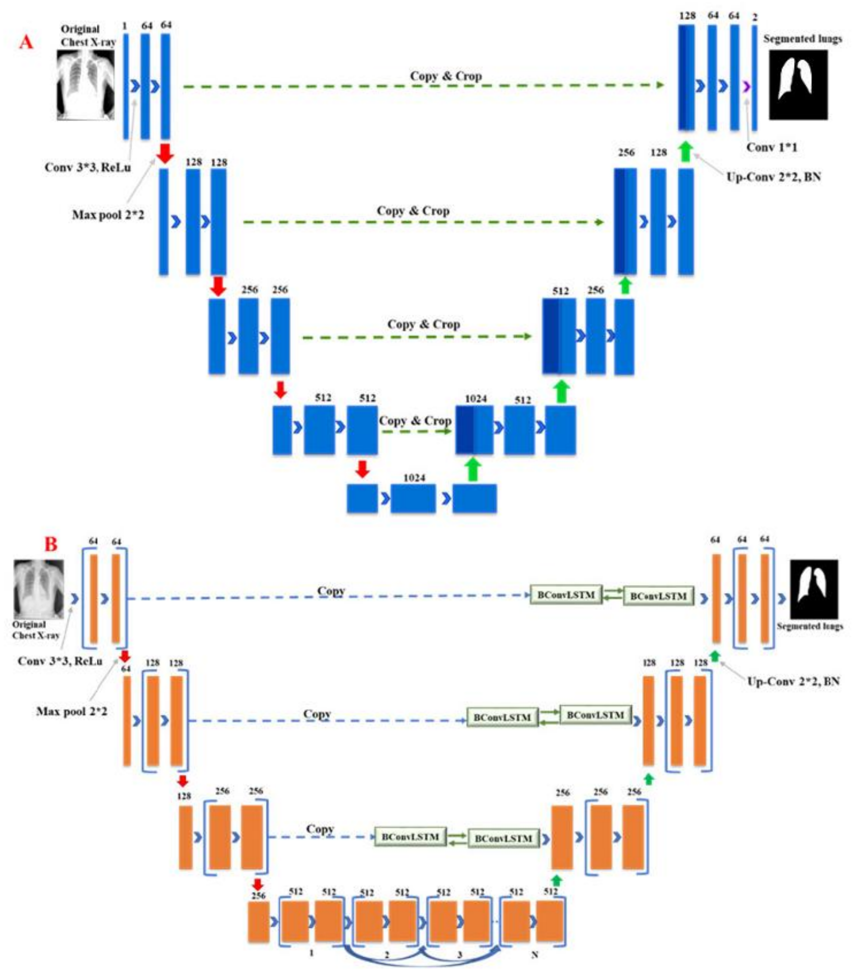


Figure 2.1.4 (b) Architecture of Original U-Net and Modified U-Net. Adapted from [22]

2. 2 Literature Matrix

Table 2.2 below presents a literature matrix that compares studies on AI-driven chest X-ray classification, highlighting their challenges, objectives, datasets, preprocessing techniques, modelling approaches, and evaluation metrics to contextualize the development of diagnostic tools for lung disease detection.

Author/ Year	Title	Problem Mentioned	Objective	Dataset	Preprocessing & Feature Engineering	Method for Modelling	Evaluation Metrics	Result
S. Sahoo, P. P. Pani, C. Dora, S. Chakravarty, 2024 [8]	Multi-Class Classification of Chest X-ray Images with Optimized Features	-Human error in manual analysis -Necessity for rapid and accurate disease identification.	-To increase diagnostic efficiency and accuracy in clinical settings.	-21,165 chest X-ray (CXR) images (Covid-19, Normal, Pneumonia) -PNG, 299×299 pixels	-Obtain features using LBP. HOG and EfficientNet B0 -Optimized using Bayesian optimization	-Then fed to the SVM and Random Forest classifier	Precision, recall, F1-score, and accuracy	SVM performed better than Random Forest with 97% accuracy and 97.34% F1-score.
Kriti Vij, Aman Sharma, Saloni Thakur, Rajni	Lung Disease Classification using X-Ray Imaging with	-Manual diagnosis can be mistake-prone.	-To speed up the detection process, enabling timely analysis	-approximately 85,000 frontal chest X-ray images -ten classes collection of X-ray images	-scaling down the images to 64x64 pixels -cropping them -transforming them into tensors	Ensemble learning using ResNet and VGG16	Precision, F1-score, Accuracy	Ensemble model achieved 97.87% precision, 97.62% accuracy.

Mohana, 2023 [9]	Ensemble Learning	-Manual interpretation takes time	and treatment of lung diseases.					
Mostofa Kamal Sagor et al., 2024 [10]	An Efficient Deep Learning Approach for Detecting Lung Disease from Chest X-Ray Images Using Transfer Learning and Ensemble Modeling	-Slow diagnostic processes -High rates of misdiagnosis due to human error.	-To make the detection process automated and faster, and also more accurate than the previous models	ChestX-Ray8 and Pneumonia datasets from Kaggle	-Resizing to 224x224x3 (scikit-image, TensorFlow, and Caffe frame works) -Scaling Images -Data Augmentation (rotation, flipping).	Utilizes transfer learning with Inception-v3, VGG16, ResNet-50 models; employs an ensemble approach for better accuracy.	Accuracy, recall, and precision across multiple deep learning models.	Ensemble model achieved 88.14% accuracy.
Rasha S. Gargees, 2022 [11]	Multi-Class Flat Classification of Lung Diseases Utilizing	-Limited availability of expert radiologists -Time-consuming	-To diagnose diseases automatically and fast -cost-saving,	Novel large multiclass dataset created from several publicly available datasets. (Covid-	-unify image sizes to 224 x 224 pixels -Image normalized to ImageNet dataset.	ResNet152, DenseNet201, ResNeXt101, SqueezeNet1_1, VGG19, and AlexNet	Accuracy, Cohen Kappa score, FBeta score, Matthews	ResNet152 model had the best performance: 95.643% accuracy

	Deep Learning	diagnostic processes -prone to diagnostic errors.	time-saving, and efficient manner	19, Lung opacity, pneumonia, tuberculosis, normal)	-split into training and validation set.		Correlation Coefficient (MCC), and Hamming Loss	
T. Ahad et al., 2024 [13]	MultiClass Classification of Chest Diseases Using CXR Images with DenseNet201+CNN and Grad CAM Visualization	-Need for early detection of lung diseases - Expensive and time-consuming COVID-19 testing (e.g., RT-PCR)	- To develop a deep learning framework for multiclass classification of chest diseases (COVID-19, pulmonary fibrosis, tuberculosis, lung opacity, normal) - To incorporate Explainable AI (Grad-CAM) for	- Custom dataset from Kaggle with 17,902 CXR images (COVID-19, pulmonary fibrosis, tuberculosis, and normal cases)	-Image Augmentation -Class Imbalance Handling -Image Resizing	- Modified DenseNet201 combined with a custom CNN - Compared with ResNet50, VGG16, VGG19, and standalone DenseNet201	Accuracy, Precision, F1 score, Recall	- Achieved 95.43% testing accuracy

			transparency in decision-making					
G. H. Dagnaw and M. E. Mouthadi, 2023 [15]	Towards Explainable Artificial Intelligence for Pneumonia and Tuberculosis Classification from Chest X-Ray	-Black-box nature of AI models raise concerns in medical imaging. - Radiologists are scarce in locations with limited resources and developing nations,	- To classify and detect tuberculosis and pneumonia using a lightweight CNN and visual-based XAI. -To enhance trust in model decisions using Score-CAM XAI.	-Total 10,965 CXR images sourced from NIAID TB portal and Kaggle (Normal, Tuberculosis, and Pneumoni)	-Images resized to 224 x 224 pixels. -Apply CLAHE for image enhancement. -Split with a ratio of 70:15:15 for training, validation, and test sets.	- Proposed lightweight CNN -Compare the proposed CNN with VGG16, Densnet201, EfficientNEtB0, InceptionV3, and MobileNetV2	Accuracy, Precision, Sensitivity, Specificity, F1-Score	-Proposed CNN achieved accuracy of 97.5%, a precision of 97.4%, a 97.4% sensitivity, a 98.6% specificity, and a 97.4% F1 score
Albahli et al., 2021 [16]	AI-driven deep CNN approach for multilabel pathology	-Previous models detected only one disease,	- To identify multiple chest-related diseases	-112,120 frontal-view CXR labeled with 14 diseases	- Reshaped images to (150,150,3), normalized, applied image augmentation	DenseNet121,InceptionReNetV2, ResNet152V2	Train/Validation Loss, Train/Validation, Accuracy,	InceptionResNetV2 achieved the highest ROC-AUC

	classification using chest X-Rays	ineffective for others. -Lack of data augmentation, highly normalized data, and ineffective hyperparameters	using deep CNNs and to tackle insufficient and unbalanced data through transfer learning.		-Used one-hot encoding for labels, reshaped to (128,128,3)		ROC-AUC Score	score of 0.801.
R. Arora, V. Bansal, H. Buckchash, R. Kumar, V. J. Sahayasheela, N. Narayanan, G. N. Pandian, and B. Raman, 2021 [17]	AI-based diagnosis of COVID-19 patients using X-ray scans with stochastic ensemble of CNNs	Need for an accurate, reliable diagnosis of COVID-19 from chest X-ray images to differentiate from other respiratory conditions.	To proposes a novel framework to classify diseased images of people using their chest X-rays among COVID 19, Normal, and Pneumonia.	- Dataset 1 (D1): total 5840 images (1575 Normal, 4265 Pneumonia) - Dataset 2 (D2): total 954 images (308 COVID-19, 323 Normal, 323 Pneumonia)	- Resized images to 224×224 pixels -Extracted 1024 features each from DenseNet and GoogLeNet.	Utilized a stochastic ensemble of CNNs - (DenseNet and GoogleNet) for feature extraction -Processing these features through a variational autoencoder -Using an ensemble of traditional ML classifiers (SVM, Random Forest, XGBoost) with logistic regression as a meta-learner	Accuracy (Ac), Sensitivity (Sen), Specificity (Spe), F1-Score, AUC.	Ensemble achieved 91.7% accuracy

Alshmrani, G. M. M. et al, 2023 [18]	A deep learning architecture for multi-class lung diseases classification using chest X-ray (CXR) images	- Challenges in accurate diagnosis of diseases like COVID-19, pneumonia, lung cancer, and TB due to similar symptoms.	- To design a deep learning framework to classify multi-class lung diseases	Total of 80,000 CXR images (Covid-19, Lung Opacity, pneumonia, lung cancer, TB, normal) from public datasets (RSNA, SIRM, Radiopaedia, etc.).	- Resized images to 224×224×3. - Normalized pixel values to [0,1]. -80% training and 20% validation.	VGG19 + CNN (three CNN blocks for feature extraction, followed by fully connected layers for classification).	- Loss -Accuracy Precision - Recall (Sensitivity- -F1-Score - AUC	-VGG19 + CNN achieved: 96.48% accuracy, 97.56% precision, 93.75% recall, 95.62% F1-score, and 99.82% AUC.
Abhishek Agnihotri, Narendra Kohli, 2023 [19]	A Hybrid Deep Neural approach for multi-class Classification of novel Corona Virus (COVID-19) using X-ray images	- Limited datasets available for COVID-19 detection.	-To analyse the performance of hybrid DL models with novel deep learning model and pretrained models	- Total of 1,512 CXR images from three public datasets: 1,212 for training (404 per class: COVID-19, Normal, Pneumonia) and 300 for validation (100 per class).	- Resized images to 224×224×3	Three proposed architectures: Novel CNN, hybrid CNN-LSTM, and hybrid CNN-RF. -Five pre-trained models: VGG-19, VGG-16, ResNet50, Inception v3, and Inception_ResNetv2.	-Accuracy -Precision -Recall -F1-Score.	CNN-RF model outperformed and achieved an accuracy of 94.66%

Rudrajit Choudhuri, Amit Paul, 2021 [20]	Multi-Class Image Classification for Detection Of Diseases Using Chest X-Ray Images	-Limited data for training models. -High risk of model overfitting on small data.	-To minimize human errors, and workload and maximize efficiency - To build a robust model to train on small available data and still have a reliable, accurate prediction in diagnosis	1368 CXR images (COVID-19, pneumonia, and normal) -Collected from 3 public sourced database.	Image normalization and resizing to 224 x 224 pixels for uniform input.	Utilized Convolutional Neural Networks (CNN) and VGG16 architecture via transfer learning	Accuracy, precision, sensitivity, and specificity	VGG16: 98.3% accuracy, precision metric of 0.94
N., Sharma, N., Saba, L., Khanna, N.N., Kalra, M.K., Fouda,	Segmentation-Based Classification on Deep Learning Model Embedded with Explainable	-Existing AI-based COVID-19 detection in CXR often lacks lung segmentation or is limited to 2-	- To develop a segmentation-based classification deep learning system for rapid,	-For Segmentation: 704 CXR images from Kaggle -For Classification: 12,926 CXR	Resizing, normalization, and segmentation using UNet and UNet+ models.	Combining UNet or UNet+ with 8 deep classifiers (VGG16, VGG19, Xception, InceptionV3, DenseNet201, NASNetMobile,	Segmentation: Accuracy, Loss, Dice, Jaccard, Area Error, AUC. - Classificatio	(UNet+Xception): 97.45% accuracy, 97.46% precision, 97.45% recall, 97.43%

M.M., Suri, J.S., 2022 [21]	AI for COVID-19 Detection in Chest X-ray Scans	3 class classification, missing high accuracy for clinical use. -Current systems fail to meet regulatory requirements (error rate <5%) for clinical settings.	precise COVID-19 detection in CXR with an error rate <5% to meet regulatory standards.	images from three datasets		ResNet50, MobileNet)	n: Accuracy, Precision, Recall, F1-Score, AUC.	F1-score, 0.998 AUC (p<0.0001) .
Rahman et al. , 2020 [22]	Reliable Tuberculosis Detection Using Chest X-Ray With Deep Learning, Segmentation, and Visualization	- Lack of trained radiologists in low-resource areas - Misclassification of TB with similar radiologic patterns	- To develop a reliable computer-aided diagnosis system for TB detection using chest X-rays	-704 images for lung segmentation - 3500 TB and 3500 normal images for classification	- Resized images to 256×256 for U-Net, 227×227 for InceptionV3, and 224×224 for other CNNs - Split data into 80% training and 20% testing with 5-fold cross-validation	- Original and Modified U-Net for lung segmentation Nine pre-trained CNNs (ResNet18, ResNet50, ResNet101, ChexNet, InceptionV3, VGG19, DenseNet201, SqueezeNet, MobileNetV2) for classification	- For segmentation: Loss, Accuracy, IoU, Dice Coefficient For classification: Accuracy, Sensitivity, Specificity, Precision, AUC, F1	- CheXNet achieved 96.47% accuracy without segmentation DenseNet201 achieved 98.6% accuracy with

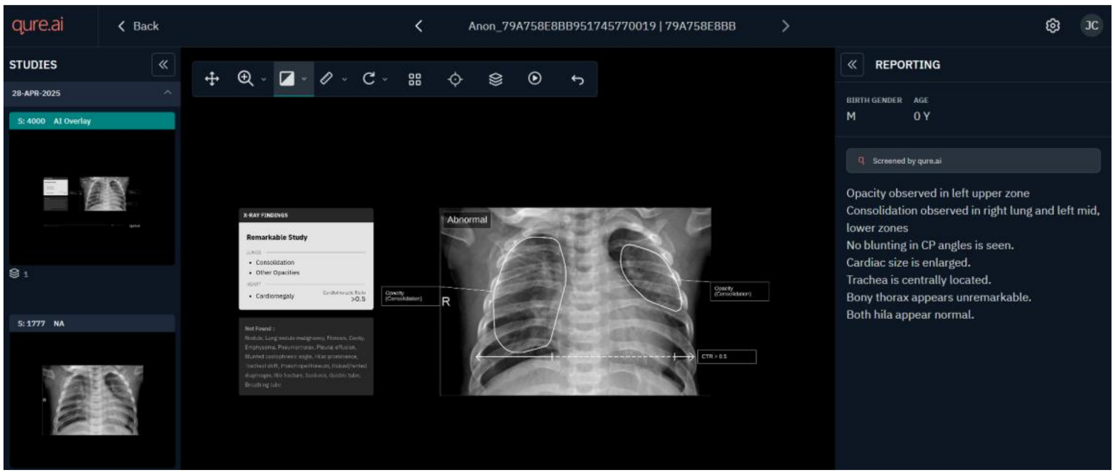
						-Score-CAM and t-SNE for visualization	Score, Processing Time, Training Time	segmentation -Segmentation improved overall performance
M. Chetoui et al., 2021 [23]	Explainable COVID-19 Detection on Chest X-rays Using an End-to-End Deep Convolutional Neural Network Architecture	-Need for distinguishing COVID-19 from other pneumonia due to similar signs	- To develop a deep learning algorithm for detecting COVID-19, pneumonia, and normal cases on CXR images	- Nine datasets -Total: 3288 COVID-19 images, 8551 normal, 5675 pneumonia for training; 1228 COVID-19, 1128 normal, 1072 pneumonia for testing	-Resized images to 512×512 -Applied class-weight approach to handle class imbalance during training	- Fine-tuned EfficientNet-B5 with Swish activation	Accuracy Sensitivity Specificity Area Under Curve (AUC)	DeepCCXR-Bin: AUC 0.985, average SN 0.94, SP 0.98

Table 2.2 Literature Matrix

2.3 Reviewing Existing Tools/System

2.3.1 Qure.ai qXR

Qure.AI is a commercial AI-powered solution developed for medical imaging analysis, as documented in research paper [24]. The company's primary product, qXR, employs deep learning algorithms to analyze chest X-rays (CXR) with remarkable accuracy. The system achieves 96% to 99% for specificity and 95% to 100% for sensitivity in lung nodule detection, while significantly decreasing interpretation time by 40.63% [24]. The qXR-HF algorithm specifically targets heart failure diagnosis by identifying key indicators like cardiomegaly and pleural effusion within 60 seconds per image. With FDA clearance for several applications and partnerships across global healthcare institutions, Qure.AI enhances radiologist workflow in clinical settings rather than replacing it. Their technology supports faster diagnosis of critical conditions including tuberculosis, lung cancer, and stroke, advancing their mission to improve healthcare accessibility worldwide [24].



X-RAY FINDINGS

Remarkable Study

LUNGS
- Consolidation
- Other Opacities

HEART
- Cardiomegaly — Cardiothoracic Ratio >0.5

Not Found :
Nodule, Lung nodule malignancy, Fibrosis, Cavity, Emphysema, Pneumothorax, Pleural effusion, Blunted costophrenic angle, Hilar prominence, Tracheal shift, Pneumoperitoneum, Raised/tented diaphragm, Rib fracture, Scoliosis, Gastric tube, Breathing tube

BIRTH GENDER	AGE
M	0 Y

Screened by qure.ai

Opacity observed in left upper zone
Consolidation observed in right lung and left mid, lower zones
No blunting in CP angles is seen.
Cardiac size is enlarged.
Trachea is centrally located.
Bony thorax appears unremarkable.
Both hila appear normal.

Figure 2.3.1 (a) : qXR Chest X-Ray Report on qure.ai

Figure 2.3.1 (a) indicates qXR Chest X-Ray Report on qure.ai. The qXR report from qure.ai indicates opacity in the left upper and lower zones and right lungs, suggesting consolidation, with no blunting of CP angles. The cardiac size is enlarged, the trachea is centrally located, and the bony thorax and hila appear unremarkable. This may point to a pulmonary infection or other pathology requiring further clinical correlation.

2.3.2 Lunit INSIGHT CXR

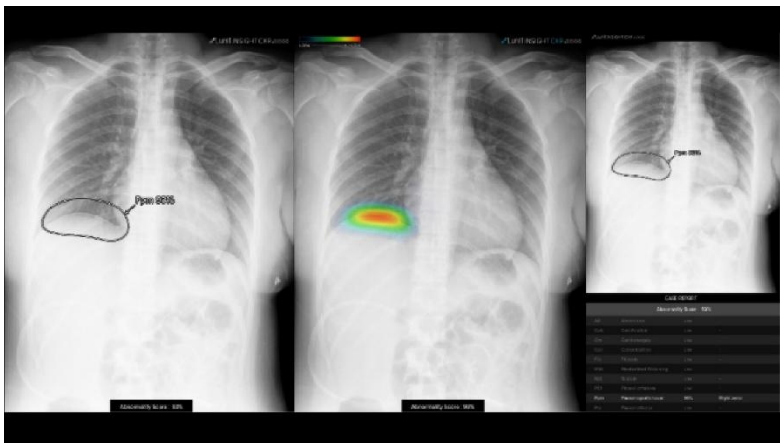


Figure 2.3.2 (a) : Chest X-Ray Report on Lunit INSIGHT CXR

Lunit INSIGHT CXR is an advanced AI-powered radiology solution that analyzes chest X-rays (CXR) to detect ten common radiologic findings with 97 to 99% accuracy (ROC AUC) [25]. The system is able to identify some lung conditions including nodules, pneumothorax, consolidation, and cardiomegaly, presenting results through visual heatmaps that pinpoint abnormalities, probability scores, and comprehensive case reports. The system has been validated through clinical testing on more than 200,000 images and bearing CE certification. This solution boosts radiologist diagnostic capability by up to 20% while decreasing interpretation time by 34% [25].

CHAPTER 3

Methodology

This section outlines the methodology and approach for developing a multi-class lung disease classification system utilizing chest X-ray (CXR) images. The process is systematically organized into distinct phases, encompassing project pre-development, data preprocessing, model training architecture design, data training, and prediction evaluation on the test dataset. The following subsections provide a comprehensive overview of the design specifications, system architecture, implementation challenges, and a detailed timeline, ensuring a structured and effective realization of the project's objectives.

3.1 Design Specifications

3.1.1 Methodologies and General Work Procedures

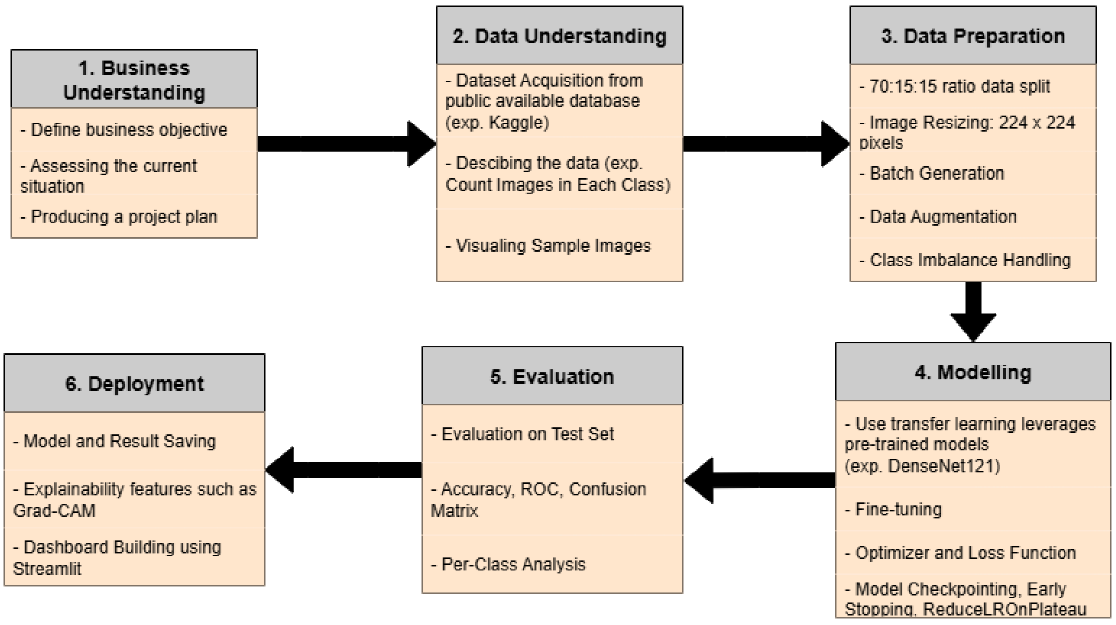

1. Business Understanding
- Define business objective
- Assessing the current situation
- Producing a project plan
2. Data Understanding
- Dataset Acquisition from public available database (exp. Kaggle)
- Descibing the data (exp. Count Images in Each Class)
- Visualing Sample Images
3. Data Preparation
- 70:15:15 ratio data split
- Image Resizing: 224 x 224 pixels
- Batch Generation
- Data Augmentation
- Class Imbalance Handling
4. Modelling
- Use transfer learning leverages pre-trained models (exp. DenseNet121)
- Fine-tuning
- Optimizer and Loss Function
- Model Checkpointing, Early Stopping, ReduceLROnPlateau
5. Evaluation
- Evaluation on Test Set
- Accuracy, ROC, Confusion Matrix
- Per-Class Analysis
6. Deployment
- Model and Result Saving
- Explainability features such as Grad-CAM
- Dashboard Building using Streamlit

Figure 3.1.1 (a) : CRISP-DM of Proposed Methodology

Figure 3.1.1 (a) indicates CRISP-DM, or the Cross-industry Standard Process for Data Mining of the methodology for realizing the multi-class lung disease classification system is planned to be structured into six distinct phases to ensure systematic development and robust validation.

Phase 1: Business Understanding begins with a foundational problem understanding phase. This project defines the objective of developing a reliable multi-class classification system capable of distinguishing between various respiratory conditions using chest X-ray images (COVID-19, pneumonia, tuberculosis, lung opacity, and normal cases). Success criteria will include target testing accuracy metrics (>90%) and performance requirements across all classes. This planning phase also includes assessing existing problems and approaches in the relative study field. Lastly, this project will design and produce the solution to achieve the goals, specify the steps to be performed during the rest of the project, including the selection of tools and techniques.

In **Phase 2: Data Understanding**, focused on data acquisition and understanding, where a publicly available chest X-ray (CXR) dataset was sourced from Kaggle that containing multiple lung diseases including COVID-19, pneumonia, tuberculosis, lung opacity, and normal cases. During this phase, a thorough data exploration was conducted to analyse the class distribution within the training, validation, and test sets. Sample images were visually inspected to assess quality variations, such as resolution and contrast, and to identify potential issues like noise or mislabelling. This process was critical for identifying challenges, particularly class imbalance, which directly informed the subsequent data preparation strategy, including the use of class weights, to ensure the dataset could support reliable multi-class classification.

Phase 3: Data Preparation involved several critical steps to prepare the dataset for model training. The dataset was systematically partitioned into a 70% training set, 15% validation set, and 15% testing set. A standard preprocessing pipeline was implemented, which included resizing all images to a consistent 224x224 pixel dimension and normalizing pixel values by scaling them to a range of [0, 1]. To address the identified class imbalance and enhance the model's generalization, data augmentation techniques were applied to the training set using Keras' ImageDataGenerator. This process included rotations (10°), zooms (10%), horizontal flipping, and minor shifts. Additionally, class weights were computed and incorporated into the training process to give a higher penalty to errors on underrepresented classes, ensuring the model's robustness and fairness across all categories.

Phase 4: Modelling centred on the development and training of a single, highly effective deep learning model: **DenseNet121**. DenseNet121: Selected for its proven efficiency and ability to enhance feature reuse through its densely connected convolutional network, enhances feature reuse by connecting each layer to every other layer, optimizing efficiency for image classification tasks [26].

Transfer learning was implemented using the DenseNet121 model pre-trained on the ImageNet dataset, which provided a robust foundation for the classification task. The training process utilized a staged fine-tuning approach with progressive layer unfreezing. Initially, the pre-trained base layers were frozen to preserve their learned features while a new custom classification head was trained. In the second stage, a subset of the base layers was unfrozen, and the entire model was fine-tuned with a lower learning rate to adapt it specifically to the chest X-ray dataset.

The model was compiled using categorical cross-entropy loss, the Adam optimizer, and accuracy as the primary performance metric. To further optimize the training, a ReduceLROnPlateau (patience=4) callback was implemented. This callback automatically reduced the learning rate when the validation loss stopped improving, which helped the model converge more effectively. Training was conducted with 20 epochs in Kaggle's GPU environment. To ensure optimal performance, early stopping (patience=5) was implemented to prevent overfitting, and model checkpointing was used to save the best weights based on validation accuracy. The final, optimized model was then saved as an .h5 file, preparing it for deployment.

Phase 5: Evaluation aims to assess model performance using comprehensive metrics, including accuracy, precision, recall, and F1-score, alongside confusion matrices for class-wise performance analysis, ROC curves, and AUC calculations to quantify discriminative capability. Visualization of model predictions on test images is planned to provide qualitative insights, ensuring the model’s reliability across all classes despite the imbalanced distribution.

Phase 6: Deployment involved integrating the final trained model into a functional web application to demonstrate its practical utility. A dashboard was developed using the Streamlit framework, providing an intuitive user interface for real-time inference. The trained DenseNet121 model (.h5 file) was loaded into the application's backend. This system allows users to upload their own chest X-ray images, which are then preprocessed and passed to the model for classification. The dashboard not only displays the model's prediction but also incorporates Grad-CAM (Gradient-weighted Class Activation Mapping) for visual interpretation. This feature highlights the specific regions of the X-ray image that the model focused on to make its classification, enhancing the transparency and interpretability of the diagnostic outcome.

3.1.2 Tools to Use

The development of the multi-class lung disease classification system leveraged a suite of specialized tools and software to ensure efficient implementation and analysis. The project was built within a Python programming environment, utilizing TensorFlow and Keras for building and training the DenseNet121 model. Data preprocessing and manipulation were handled using libraries such as NumPy and OpenCV, while Keras's ImageDataGenerator facilitated data augmentation. The entire training process was executed on Kaggle's GPU environment, specifically utilizing a NVIDIA Tesla T4 GPU. Visualization of evaluation metrics, including confusion matrices and performance plots, was achieved using Matplotlib and Seaborn. For the deployment phase, the interactive dashboard was developed in Visual Studio Code using the Streamlit framework. Overall, this combination of tools enabled a comprehensive and reproducible development workflow.

3.1.3 System Performance Definition

The performance of the multi-class lung disease classification system was defined by a set of specific and quantitative targets to ensure its effectiveness. To address the inherent class imbalance within the dataset, the system's performance was evaluated using multiple metrics beyond simple accuracy. The project aimed to achieve a minimum accuracy of 90% on the test dataset, with precision, recall, and F1-score targeted to exceed 85% for each class to ensure a balanced performance. The discriminative capability of the model was further validated by targeting AUC values above 0.90 from the ROC curves. The system's efficiency was also a key consideration, with inference time measured to ensure rapid and practical application, a process facilitated by Kaggle's GPU resources. These metrics were used to objectively assess the final model's reliability and effectiveness across all categories.

3.1.4 Verification Plan

The verification plan for the multi-class lung disease classification system was executed to validate its functionality and performance. The system was rigorously tested using a designated test dataset of CXR images, which included a representative sample of all classes. The initial verification process computed key performance metrics, including accuracy, precision, recall, and F1-score, which were compared against the ground truth labels to ensure alignment with the established targets. A confusion matrix was also generated to provide a detailed analysis of the model's class-wise performance.

A separate, in-depth verification was conducted to assess the model's real-world practicality and interpretability. This process utilized a dedicated notebook (local_model_investigation.ipynb) to analyse the model on individual test images. This analysis specifically included a Grad-CAM verification to calculate a Clinical Relevance Rate, which provided evidence that the model was focusing on the correct lung regions. A final Deployment Score was also computed, which served as a comprehensive measure to validate the model's readiness for its final implementation. This multi-faceted approach provided both quantitative and qualitative validation of the system's reliability and transparency.

3.2 Implementation Issues and Challenges

The implementation phase of the project encountered several technical and clinical challenges that required careful consideration. From a data perspective, the significant class imbalance within the dataset was a primary hurdle, as it could skew model predictions toward the majority classes. This necessitated the implementation of effective countermeasures, such as class weighting and a strategic approach to model training. Computationally, training a deep learning model like DenseNet121 was intensive, and the constraints of Kaggle's free GPU environment limited extensive experimentation with hyperparameter tuning. From a clinical standpoint, the radiological similarities between different respiratory diseases, such as COVID-19 and other pneumonias, presented a fundamental classification challenge, requiring a robust model that could distinguish subtle differences in opacity patterns.

The project introduced several novel aspects to address these challenges and enhance the system's utility. A staged fine-tuning approach with progressive layer unfreezing was implemented to optimize the DenseNet121 model's performance on the specific dataset. For clinical interpretability, Gradient-weighted Class Activation Mapping (Grad-CAM) was integrated into the system's verification and dashboard. This technique generates heatmaps that visually highlight the regions of the X-ray image most influential in the model's decision-making process. This feature addresses a critical gap in current diagnostic tools by providing visual confirmation of the model's focus areas, thereby building trust and moving the system beyond a black-box model towards a deployable diagnostic support tool.

3.3 Timeline

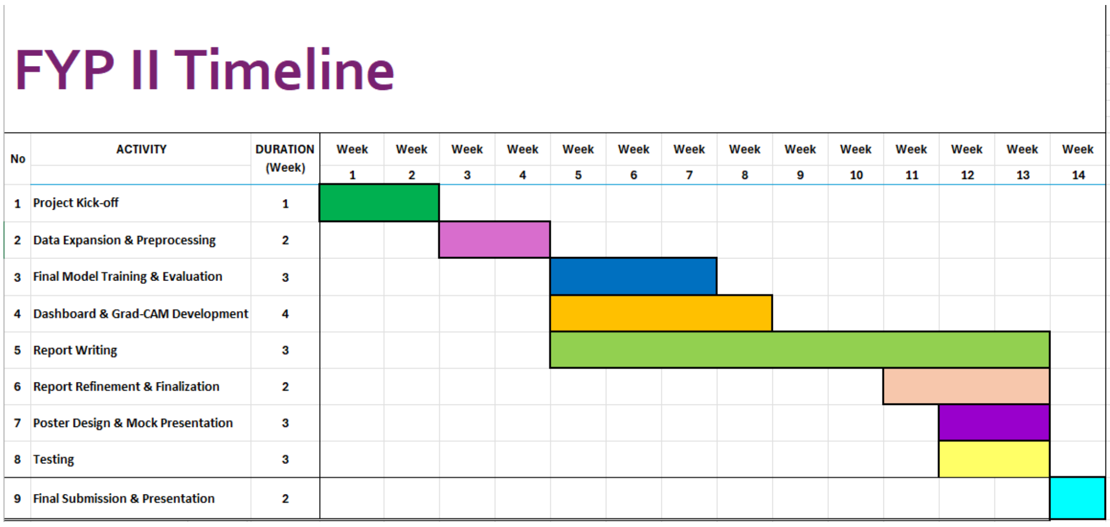


Figure 3.3 (a) :Gantt Chart of FYP II

The Final Year Project II, spanning from June to October 2025, focused on transforming the initial prototype into a fully functional and deployable system. The first key activity involved data expansion, where the existing dataset was enhanced to improve the model's robustness. This was followed by a phase dedicated to model fine-tuning and re-training of the DenseNet121 architecture. A major component of the project was the dashboard development, which involved building the user interface in Streamlit and integrating the final model. During this time, crucial features like Grad-CAM visualization were also developed and implemented to provide clinical interpretability. The final months were dedicated to comprehensive evaluation and testing to ensure the system's reliability, culminating in final report writing and poster design for the final project presentation.

CHAPTER 4

System Design

This chapter explores the architectural design of the AI-driven chest X-ray diagnostic system, detailing the systematic approach to developing a multi-functional medical support dashboard. Through activity diagrams, use case scenarios, and interface wireframes, the chapter illustrates the strategic integration of advanced AI technology with practical healthcare solutions, emphasizing user-centric design and comprehensive diagnostic support.

4.1 Activity Diagram

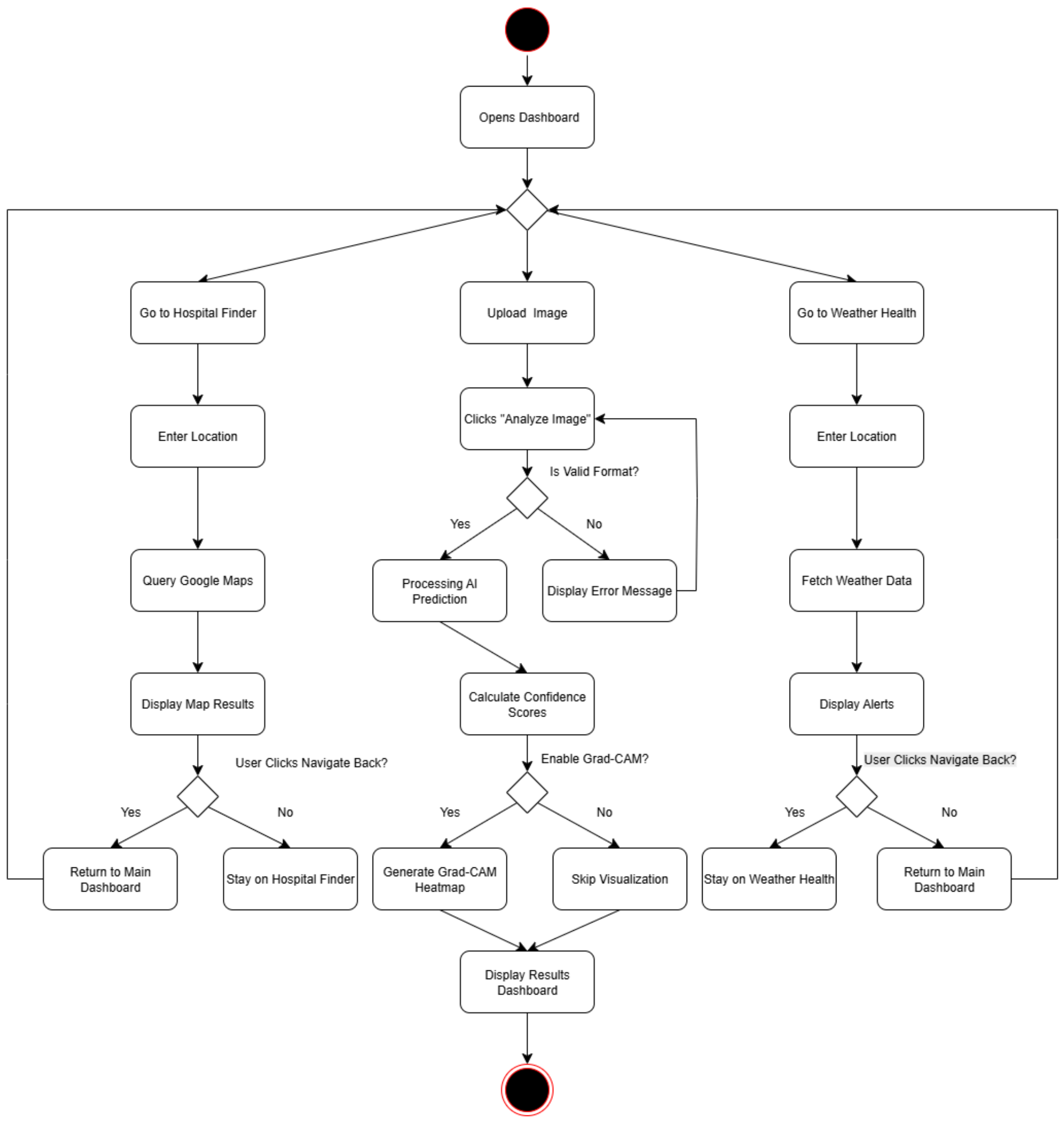


Figure 4.1 (a) : Activity Diagram

1. Homepage: The homepage design centers on a user-friendly medical image diagnostic workflow. The interface allows seamless chest X-ray image upload with comprehensive validation mechanisms to ensure only appropriate medical images are processed. The core logic involves preprocessing the uploaded image, passing it through the DenseNet121 model, and generating a detailed diagnostic report with confidence-based recommendations. The system incorporates multiple safety checks, including file format validation, image quality assessment, and AI prediction confidence thresholds to maintain high diagnostic reliability.

2. Hospital Finder: The Hospital Finder page provides a location-based medical facility search functionality. Users can input their location to receive an interactive Google Maps integration displaying nearby hospitals and medical centers. The design focuses on quick, accessible emergency medical information retrieval, with a streamlined interface that offers direct navigation, contact details, and contextual healthcare location guidance. The workflow emphasizes user-friendly navigation and immediate access to critical medical infrastructure information.

3. Weather Health: The Weather Health feature delivers comprehensive environmental health monitoring through real-time data integration. By leveraging the OpenWeatherMap API, the page generates localized weather and air quality insights, presenting users with actionable health recommendations based on current environmental conditions. The design prioritizes presenting complex meteorological and air quality data in an intuitive, visually engaging format, with specific attention to respiratory health considerations for individuals with existing medical conditions.

4.2 Use Case Diagram and Description

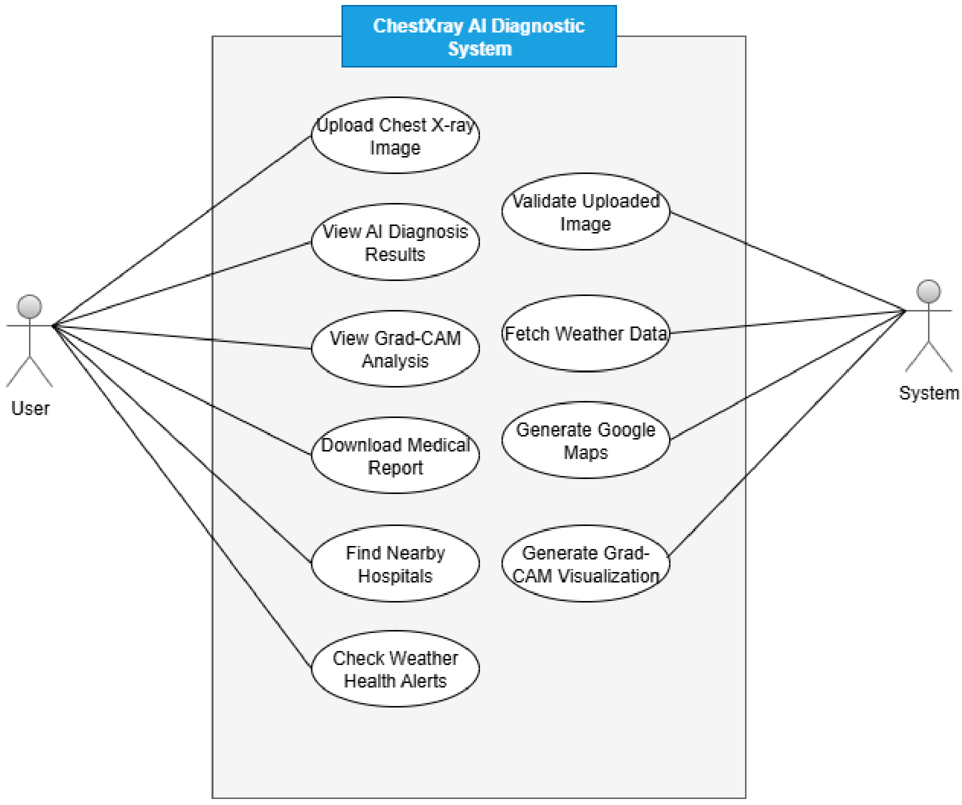


Figure 4.2 (a) : Use Case Diagram

User Interaction Design: The user interface is designed to provide intuitive and comprehensive access to medical diagnostic and health support features. Users can seamlessly upload chest X-ray images for AI analysis, view detailed diagnostic results with confidence scores, explore Grad-CAM visualizations for model interpretability, and download comprehensive medical reports. Additionally, the system offers supplementary functionalities like finding nearby hospitals and accessing localized weather health alerts, creating a holistic healthcare support platform that empowers users with actionable medical insights.

System Backend Architecture: The system's backend is engineered to deliver robust, multi-layered functionality through sophisticated technical integrations. It employs advanced image validation mechanisms to ensure diagnostic accuracy, leverages the DenseNet121 model for precise medical image classification, and integrates external APIs like Google Maps and OpenWeatherMap to provide contextual health information. The backend processes include automatic image preprocessing, AI prediction generation, Grad-CAM heatmap visualization, and real-time data retrieval, all designed to maintain high standards of computational efficiency, data security, and user-centric information delivery.

4.3 Wireframe

Wireframe - Homepage (Upload CXR)

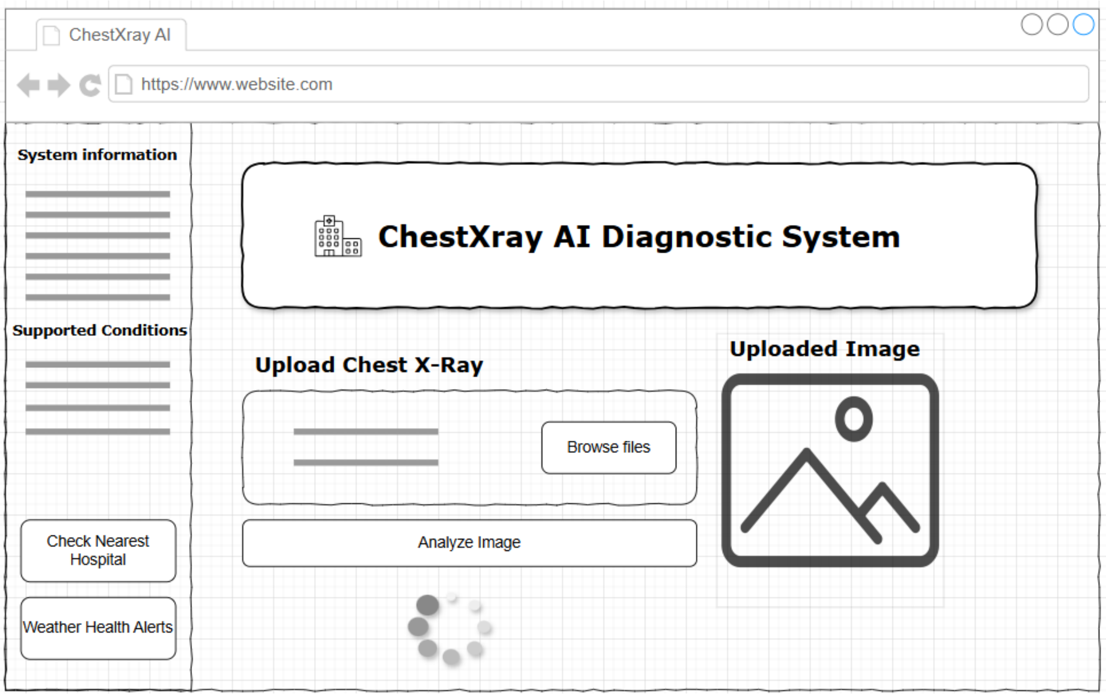


Figure 4.3 (a) : Wireframe - Homepage (Upload CXR)

The homepage wireframe shown in Figure 4.3 (a) indicates that the wireframe for chest X-ray upload is designed with a focus on medical interface usability and user experience. The interface provides an intuitive, single-task approach to image upload, allowing users to easily drag and drop or browse chest X-ray images. Clear visual feedback and error prevention mechanisms guide users through the upload process, ensuring only appropriate medical images are processed.

The design prioritizes accessibility and clarity, with a clean, straightforward layout that reduces cognitive load. Touch-friendly elements support mobile and tablet interactions, while progressive loading states and persistent session management ensure a smooth user experience. The wireframe emphasizes immediate, descriptive feedback about file compatibility and upload status, creating a transparent and user-friendly diagnostic support interface

Wireframe - Homepage (Result)

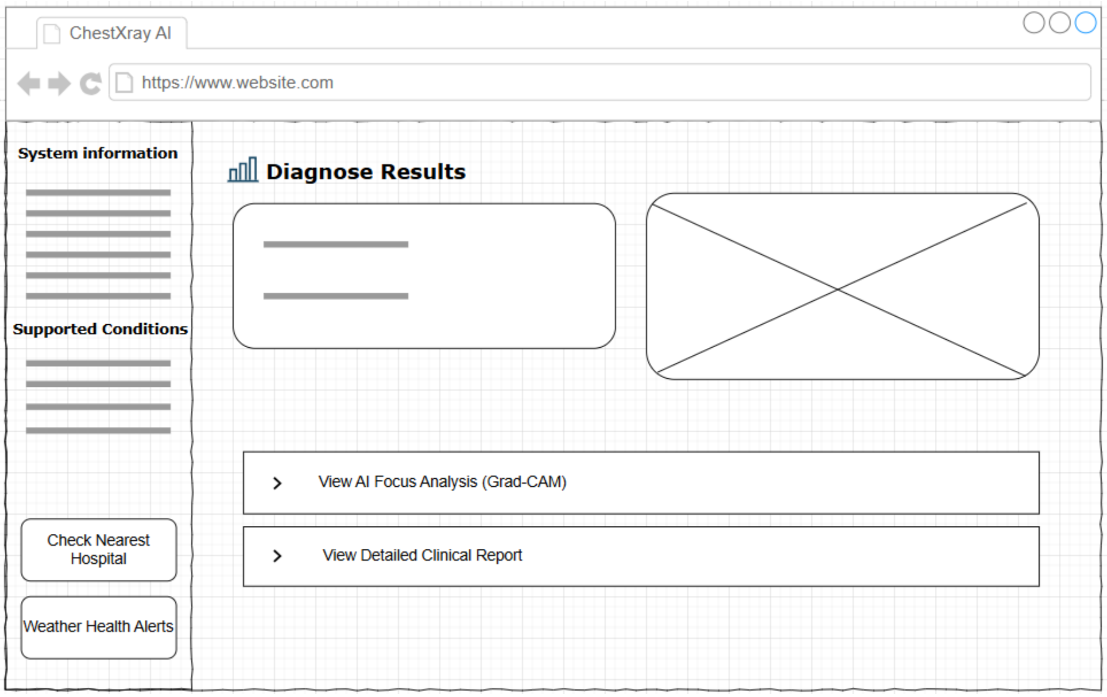


Figure 4.3 (b) : Wireframe - Homepage (Result)

The result page wireframe presents medical diagnostic information with a clear, hierarchical design prioritizing clinical insights. A color-coded confidence indicator provides instant visual comprehension, with primary diagnosis prominently displayed and confidence metrics immediately visible. Interactive elements balance information density with user-friendly navigation, allowing progressive access to detailed information.

The interface emphasizes transparency through professional disclaimers, warning systems for low-confidence predictions, and time-stamped results formatted for clinical documentation. Expandable sections and integrated tooltips help users understand complex medical terminology while maintaining a clean, focused design.

Wireframe - Hospital Finder

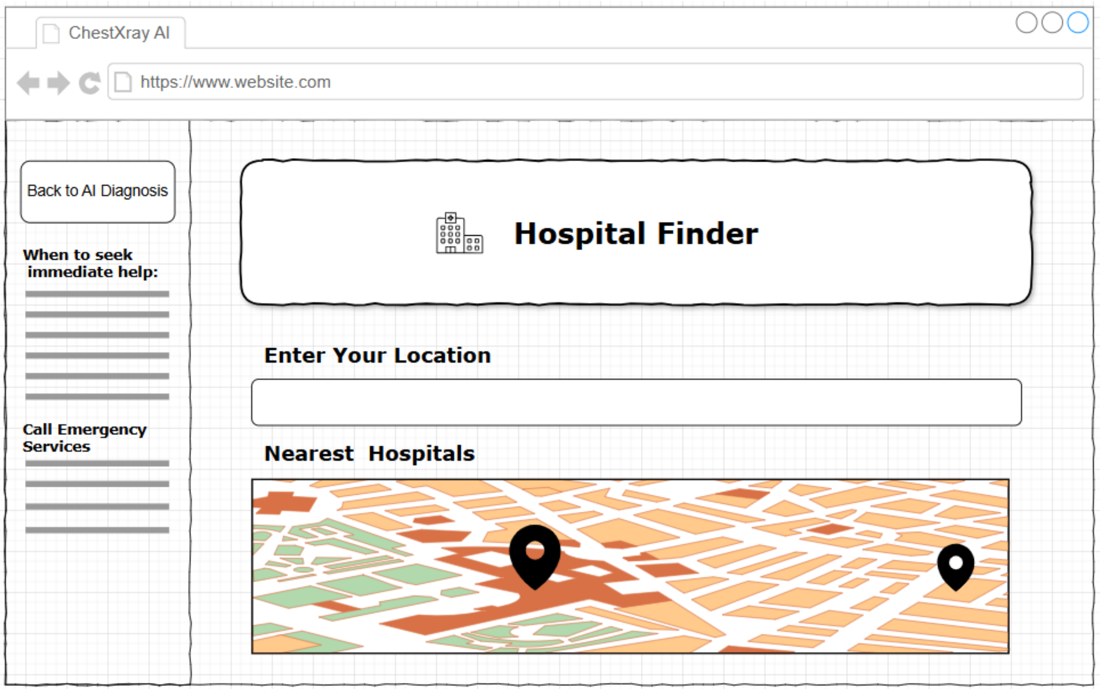


Figure 4.3 (c) : Wireframe – Hospital Finder

The Hospital Finder wireframe is designed with an emergency-first approach, prioritizing rapid and intuitive access to critical medical facilities. The interface features a simple, single-step location entry mechanism that immediately triggers an interactive map display. Emergency contact information, including universal emergency numbers, remains constantly visible, with urgent situation alerts prominently highlighted in red to ensure immediate user attention.

The geographic information interface centers on a large, interactive map that dynamically updates with location-based hospital information. Intelligent design features include automatic search triggering after minimal location input, clear proximity markers, and touch-friendly elements optimized for mobile and one-handed operation. The information architecture follows a progressive disclosure model, guiding users from location entry through map visualization to detailed facility information, with built-in context-sensitive help and clear navigation options back to the main diagnostic system.

Wireframe - Weather Health Alerts

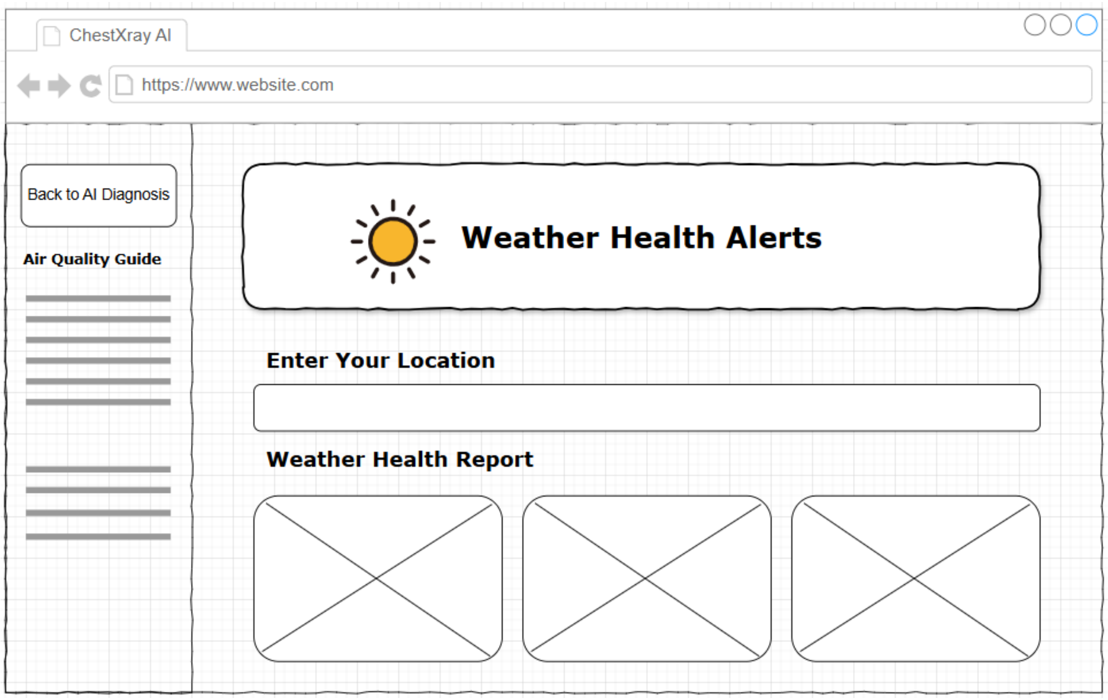


Figure 4.3 (d) : Wireframe – Weather Health Alerts

The Weather Health Alerts wireframe transforms standard weather information into actionable medical guidance. By presenting real-time environmental data through intuitive, color-coded icons, the interface provides immediate insights into potential health risks, with a specific focus on respiratory and lung health.

The design offers mobile-optimized, patient-centered recommendations that translate complex meteorological data into clear, actionable health insights. Leveraging WHO guidelines, the wireframe helps users quickly understand environmental health risks and their potential impact on medical conditions.

CHAPTER 5

System Implementation

This chapter provides a comprehensive overview of the system's implementation, detailing the technical environment and key development stages. It begins with an outline of the Hardware Setup (Section 5.1) and Software & Library Setup (Section 5.2), which define the project's computational and coding environments. The process of model training is then covered in Training and Configuration (Section 5.3), describing the hyperparameters and environment used. The chapter concludes by showcasing the functionality of the system in System Operation and Deployment (Section 5.4) and discussing the technical Implementation Issues and Challenges (Section 5.5) that were encountered.

5.1 Hardware Setup

Laptop

Item	***Specification***
Model	HP Pavilion Laptop 15-eg3xxx
System Type	64-bit operating system, x64-based PC
Processer	13th Gen Intel(R) Core(TM) i5-1335U, 1300 Mhz, 10 Core(s), 12 Logical Processor(s)
Graphics Card	Intel ® Iris ® Xe Graphics
Installed RAM	16.0 GB
Operating Name	Microsoft Windows 11 Home Single Language

Table 5.1 Hardware Setup of the PC

5.2 Software & Library Setup

Software/ Library	***Description***
Kaggle	A cloud-based notebook environment that provides free access to powerful GPUs for training machine learning models.
Visual Studio Code	A lightweight but powerful source code editor used for writing, debugging, and managing the project's code.
Google Colab	A cloud-based notebook environment similar to Kaggle, used for running and developing Python code directly in the browser with free access to GPUs
Google Maps API	A set of web services that provide location-based information, which can be used for integrating maps and geographical data into an application.
OpenWeatherMap	A service that provides weather data and forecasts through an API, used for integrating real-time weather information into a project.
Streamlit (1.49.1)	A Python library used for building and sharing custom web applications and interactive dashboards for data science projects.
Python (3.13.2)	The primary programming language used for developing the entire project, from model training to deployment.
TensorFlow (2.20.0)	A powerful open-source machine learning framework used for building and training the deep neural network model.
NumPy (2.2.6)	A fundamental Python library for high-performance numerical operations, essential for handling large datasets and arrays.
Matplotlib (3.10.6)	A comprehensive library for creating static, animated, and interactive visualizations in Python, used for plotting training results.
Scikit-learn (1.7.2)	A popular machine learning library used for various tasks, including data preprocessing, model selection, and performance metric calculation.

Table 5.2 Software and Library Setup

Kaggle

Figure 5.2.1 : Kaggle Logo

Kaggle is a widely used cloud-based platform for data science and machine learning that provides an integrated development environment for data scientists and researchers. The platform offers free access to computational resources including CPUs, TPUs, and GPUs, pre-installed data science libraries, and collaborative notebook environments similar to Jupyter. Kaggle provides essential features such as dataset hosting and sharing, competition hosting for machine learning challenges, community collaboration tools, and most importantly, free GPU acceleration for training deep learning models. Users can create, share, and execute Python and R notebooks directly in the browser without requiring local installation of complex machine learning frameworks.

In this project, Kaggle serves as the primary training environment for developing the chest X-ray classification model due to its provision of free Tesla T4 GPU access with up to 30 hours of weekly usage. The GPU acceleration was essential for training the deep learning model on the large chest X-ray dataset, as local CPU training would have been computationally prohibitive and time-consuming. Kaggle's pre-configured environment with TensorFlow, Keras, and other machine learning libraries eliminated the need for complex local setup, allowing immediate access to the required frameworks. The platform enabled efficient model training through its notebook interface, automatic saving of model outputs, and seamless integration with the dataset, ultimately facilitating the development of the progressive unfreezing training strategy implemented in my notebook.

Google Colab

Figure 5.2.2 : Google Colab Logo

Google Colab is a free cloud-based Jupyter notebook environment provided by Google that enables users to write and execute Python code directly in the browser without any local setup requirements. The platform offers free access to computational resources including CPUs, GPUs (Tesla T4, K80), and TPUs, along with pre-installed popular data science and machine learning libraries such as TensorFlow, PyTorch, Keras, and scikit-learn. Google Colab provides seamless integration with Google Drive for file storage, supports real-time collaboration, and offers both free and paid tiers with enhanced resource allocation for intensive machine learning workloads.

In this project, Google Colab served as a backup training environment after Kaggle, providing additional GPU resources when Kaggle's 30-hour weekly GPU limitations were exceeded during the DenseNet121 model development phase. The platform's free Tesla T4 GPU access ensured uninterrupted model training and experimentation, particularly during the progressive unfreezing strategy implementation. Google Colab's pre-configured TensorFlow and Keras environment maintained compatibility with the model architecture developed in Kaggle, allowing seamless transfer of training workflows and ensuring consistent progress in the chest X-ray classification model development.

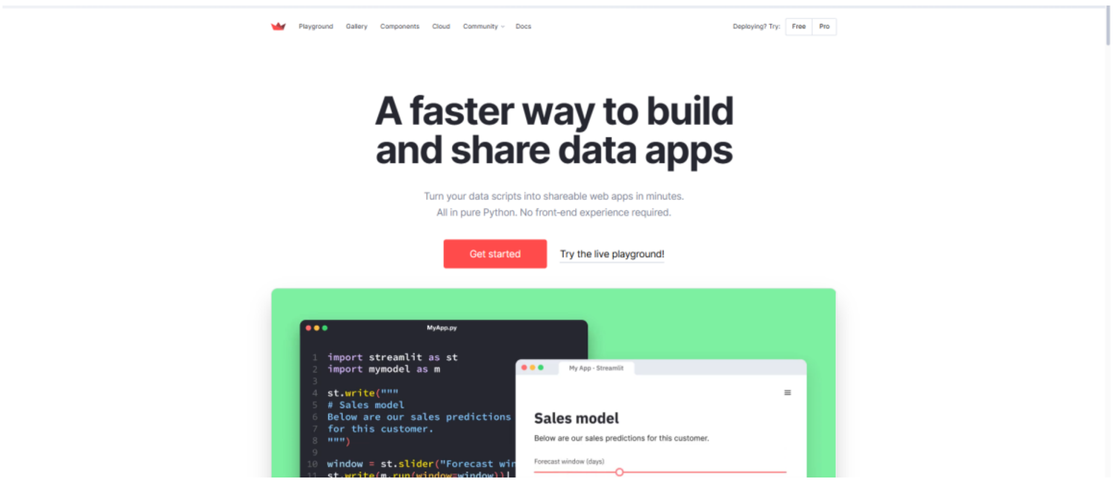
Visual Studio Code

Figure 5.2.3 : Visual Studio Code Logo

Visual Studio Code (VS Code) is a free, open-source code editor developed by Microsoft that provides a lightweight yet powerful development environment for various programming languages, including Python. The editor features built-in support for syntax highlighting, intelligent code completion, debugging capabilities, and an extensive marketplace of extensions that enhance functionality for specific development needs. VS Code offers integrated terminal access, Git version control integration, workspace management, and seamless file navigation, making it particularly suitable for Python development with extensions like Python IntelliSense, Jupyter notebook support, and automated code formatting tools.

In this project, Visual Studio Code served as the primary development environment for building the Streamlit dashboard application, providing an efficient platform for writing and managing the Python web application code. The editor's Python extension enabled intelligent code completion and error detection during the development of dashboard components including the main application interface, weather health alerts, and hospital finder functionality. VS Code's integrated terminal facilitated easy testing of the Streamlit application locally, while its file management capabilities streamlined the organization of multiple Python modules, configuration files, and static assets within the dashboard directory structure.

Streamlit



A faster way to build and share data apps

Turn your data scripts into shareable web apps in minutes.
All in pure Python. No front-end experience required.

```
import streamlit as st
import mymodel as m

st.write("""
# Sales model
Below are our sales predictions
for this customer.
""")

window = st.slider("Forecast wi
st.write(m.run(window=window))
```

Sales model

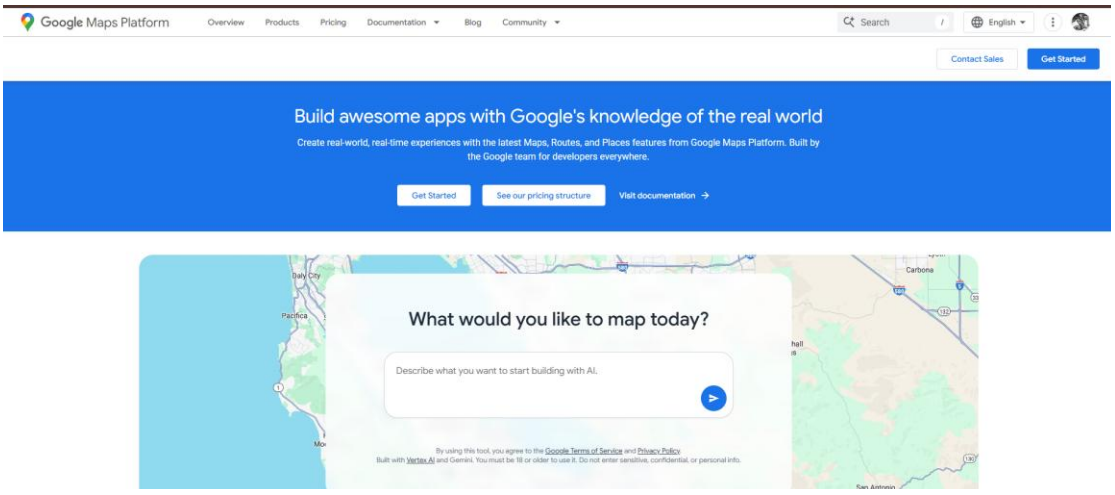
Below are our sales predictions for this customer.

Figure 5.2.4 : Streamlit

Streamlit is a Python framework that enables developers to create interactive web applications for data science and machine learning projects with minimal code. It transforms Python scripts into shareable web apps, featuring automatic updates when user inputs change, built-in widgets, and native support for data visualization libraries. Streamlit provides essential components like file uploads, forms, charts, and maps, making it a powerful tool for building professional-looking applications without extensive web development knowledge.

In this project, Streamlit serves as the core web application framework for developing the chest X-ray AI diagnostic dashboard, providing the foundation for creating an interactive user interface that integrates machine learning predictions with health monitoring features. The framework enabled the development of key functionalities including image upload and processing for chest X-ray analysis, real-time display of AI diagnosis results with confidence scores, and interactive visualization of Grad-CAM interpretability maps. Streamlit's reactive architecture facilitated the integration of external APIs for weather health alerts and hospital finder features, while its session state management ensured smooth user navigation between different dashboard pages and maintained uploaded image data throughout the diagnostic workflow.

Google Maps API



Build awesome apps with Google's knowledge of the real world

Create real-world, real-time experiences with the latest Maps, Routes, and Places features from Google Maps Platform. Built by the Google team for developers everywhere.

What would you like to map today?

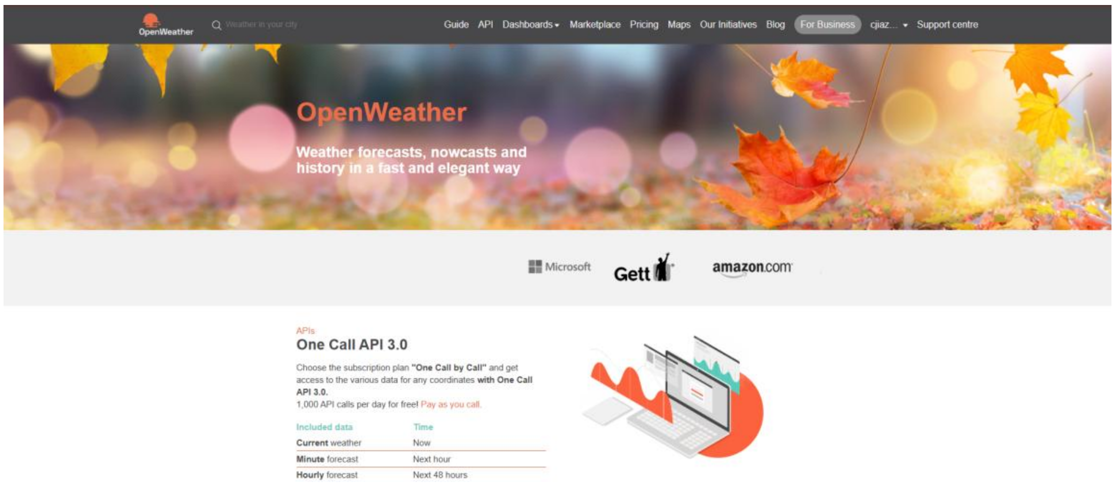
By using this tool, you agree to the Google Terms of Service and Privacy Policy. Built with Vertex AI and Gemini. You must be 18 or older to use it. Do not enter sensitive, confidential, or personal info.

Figure 5.2.5 : Google Maps API

Google Maps API is a collection of web services that allows developers to integrate Google Maps functionality into their applications. It provides services for displaying maps, searching for locations, and converting addresses into coordinates. The API supports various programming languages and has both free and paid plans, making it accessible for a wide range of applications.

In this project, the Google Maps API functions as the backend for the hospital finder feature within the AI diagnostic dashboard. It enables users to locate nearby medical facilities based on their location. The Maps Embed API specifically displays an interactive map showing hospital search results, allowing users to input their city or address to get real-time recommendations. This integration improves the diagnostic workflow by providing immediate access to emergency medical services when concerning chest X-ray results are detected.

OpenWeatherMap API



APIs

One Call API 3.0

Choose the subscription plan **"One Call by Call"** and get access to the various data for any coordinates **with One Call API 3.0.**
1,000 API calls per day for free! Pay as you call.

Included data	Time
Current weather	Now
Minute forecast	Next hour
Hourly forecast	Next 48 hours

Figure 5.2.6 : OpenWeatherMap

OpenWeatherMap API is a weather data service that provides real-time and historical weather information, including current conditions and forecasts. It offers various weather metrics like temperature, humidity, and air pollution indices. The API has a free tier for developers and supports different data formats, making it accessible for a wide range of applications.

In this project, the OpenWeatherMap API powers the weather health alerts feature in the dashboard, providing location-based environmental data relevant to respiratory health. It delivers current air quality measurements like PM2.5 and PM10 particle levels, which are critical for patients with lung conditions. This integration allows the dashboard to provide personalized health recommendations, such as advising users to limit outdoor activities during periods of poor air quality.

TensorFlow

Figure 5.2.7 : TensorFlow Logo

TensorFlow is a comprehensive open-source machine learning framework that provides tools for building, training, and deploying deep learning models. Developed by Google, it offers high-level APIs like Keras for fast development, supports various neural network architectures, and uses GPU acceleration for intensive training. In this project, TensorFlow served as the core framework for developing the DenseNet121 model, enabling the implementation of transfer learning and Grad-CAM features. Its integration with Keras facilitated efficient model design, and its GPU support was crucial for the intensive training needed for the medical image classification task.

Matplotlib

matplotlib

Figure 5.2.8 : Matplotlib Logo

Matplotlib is a comprehensive 2D plotting library for Python that provides static, animated, and interactive visualizations. It offers extensive customization for creating publication-quality plots such as line graphs, bar charts, and heatmaps. In this project, Matplotlib was the primary tool for analyzing model training performance and evaluation metrics, generating essential plots like accuracy curves, loss graphs, and confusion matrices. Its integration with TensorFlow and scikit-learn allowed for a detailed visualization of training strategies and a comparative analysis of model performance.

5.3 Training and Configuration

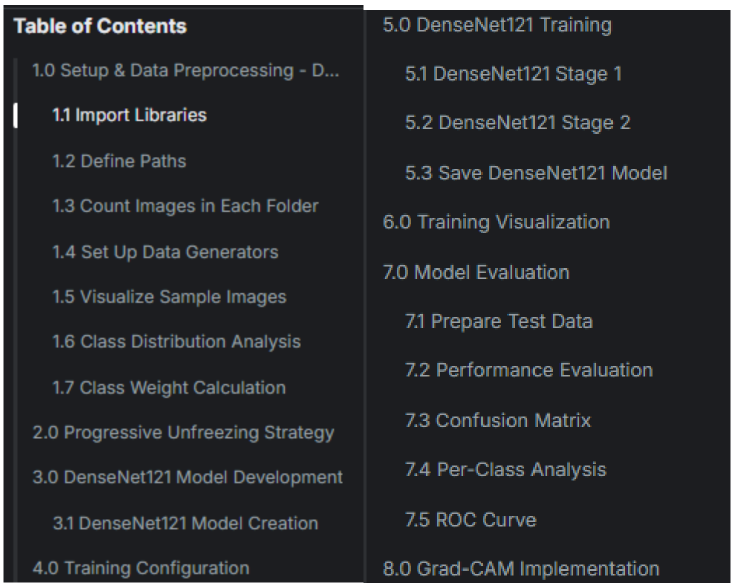


Table of Contents

1.0 Setup & Data Preprocessing - D...
1.1 Import Libraries
1.2 Define Paths
1.3 Count Images in Each Folder
1.4 Set Up Data Generators
1.5 Visualize Sample Images
1.6 Class Distribution Analysis
1.7 Class Weight Calculation
2.0 Progressive Unfreezing Strategy
3.0 DenseNet121 Model Development
3.1 DenseNet121 Model Creation
4.0 Training Configuration
5.0 DenseNet121 Training
5.1 DenseNet121 Stage 1
5.2 DenseNet121 Stage 2
5.3 Save DenseNet121 Model
6.0 Training Visualization
7.0 Model Evaluation
7.1 Prepare Test Data
7.2 Performance Evaluation
7.3 Confusion Matrix
7.4 Per-Class Analysis
7.5 ROC Curve
8.0 Grad-CAM Implementation

Figure 5.3. (a): Chest X-ray Classification Training Process

This project implemented a comprehensive approach to deep learning model development using the DenseNet121 architecture, featuring a structured workflow from data preprocessing to advanced model evaluation. The methodology systematically covers critical stages including data preparation, image analysis, model creation, and training, employing a progressive unfreezing strategy and rigorous evaluation techniques. Through detailed steps ranging from initial dataset examination to performance assessment, confusion matrix analysis, and Grad-CAM implementation, the project demonstrates a methodical approach to developing and validating deep learning models.

5.3.1 Setting Up Working Environment

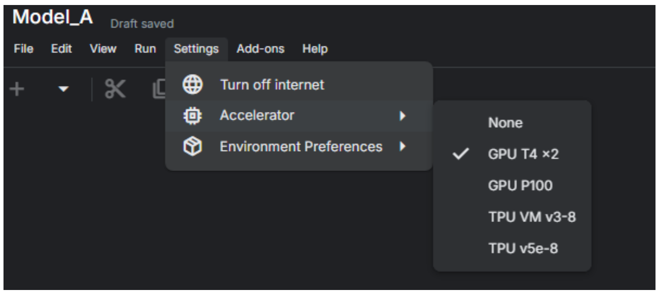


Figure 5.3 (b) : Turn On Accelerator

The runtime environment was configured to utilize GPU acceleration, specifically enabling the T4 x2 GPU option. This configuration provides computational resources for efficient model training, leveraging parallel processing capabilities to optimize computational performance during the deep learning workflow.

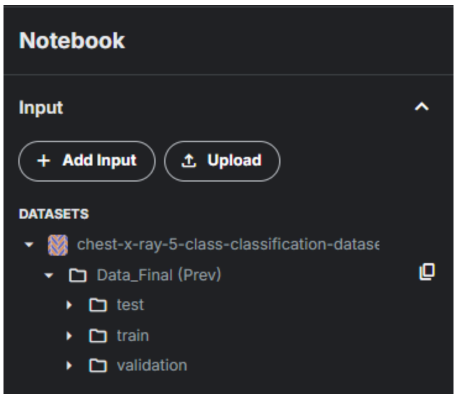
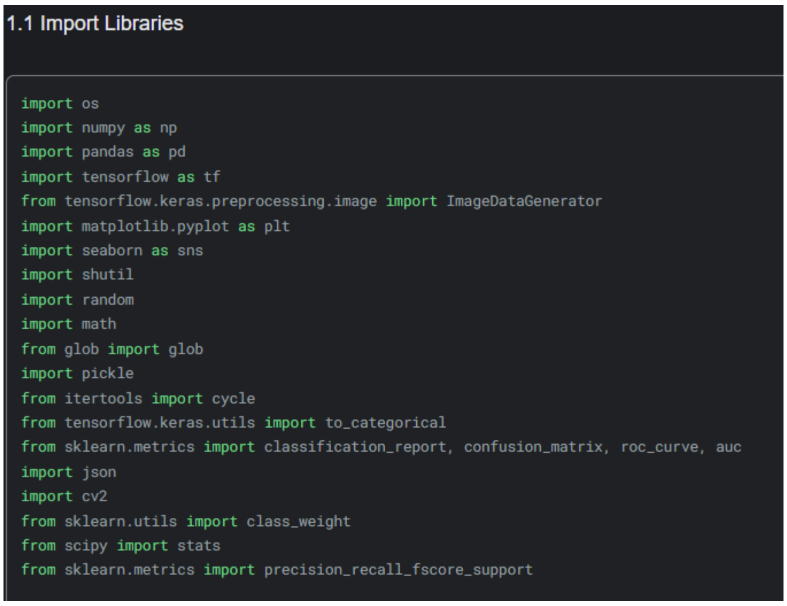


Figure 5.3 (c) : Upload Dataset

The chest X-ray (CXR) dataset was uploaded to the Kaggle notebook, comprising a comprehensive multi-class classification dataset stratified into train, test, and validation subsets. This dataset encompasses five distinct pathological classes: COVID-19, pneumonia, tuberculosis, lung opacity, and normal cases. The structured division of data into separate sets ensures robust model training, validation, and testing, facilitating a rigorous evaluation of the deep learning model's diagnostic performance across different medical imaging scenarios.

The project utilizes the following essential libraries for the deep learning pipeline:

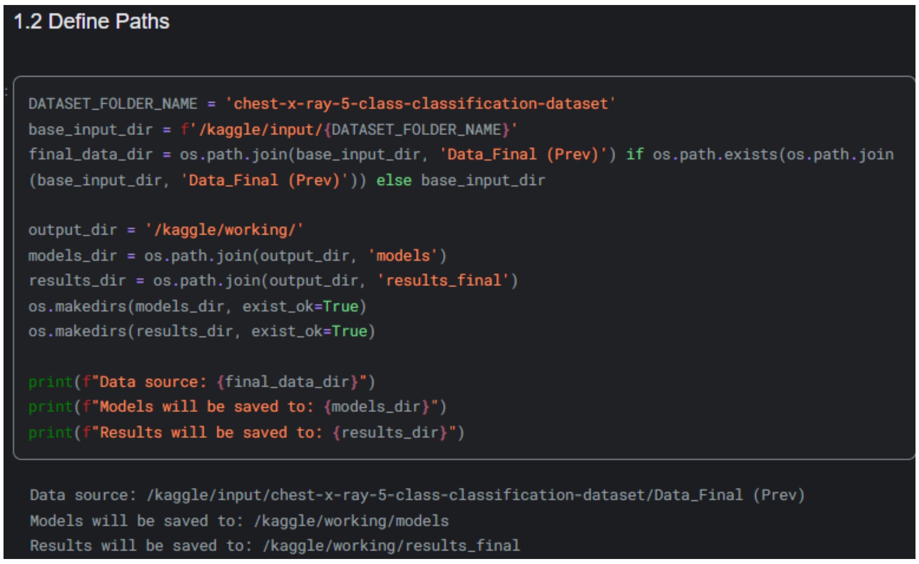


1.1 Import Libraries

```python
import os
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import seaborn as sns
import shutil
import random
import math
from glob import glob
import pickle
from itertools import cycle
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
import json
import cv2
from sklearn.utils import class_weight
from scipy import stats
from sklearn.metrics import precision_recall_fscore_support
```

Figure 5.3 (d) : Import Libraries

Figure 5.3 (d) above indicates the step of importing all necessary Python libraries and modules required for the chest X-ray classification model development. The imports include core data manipulation libraries (**NumPy** and **Pandas**), the **TensorFlow** deep learning framework with **Keras** preprocessing utilities, and visualization libraries (**Matplotlib** and **Seaborn**) for data analysis and results presentation. Machine learning evaluation tools from **scikit-learn** provide essential metrics for model assessment, while **OpenCV** handles advanced image processing operations. Additional utilities include file system operations (**os**, **glob**), data serialization (**pickle**, **json**), and statistical analysis tools (**scipy.stats**) that support the comprehensive model training and evaluation pipeline implemented throughout the notebook.


1.2 Define Paths

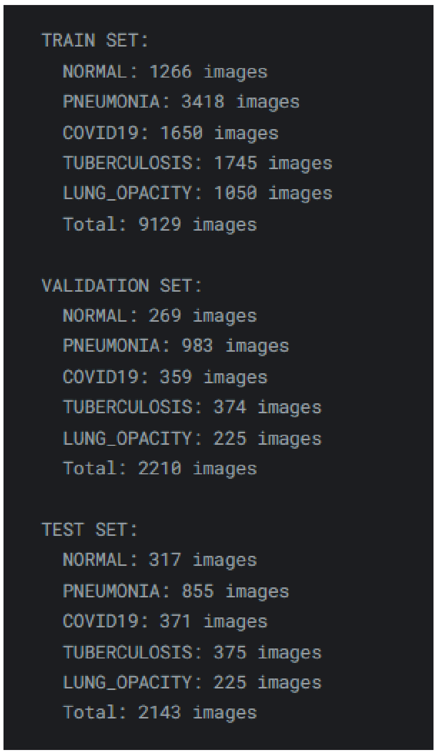
```
DATASET_FOLDER_NAME = 'chest-x-ray-5-class-classification-dataset'
base_input_dir = f'/kaggle/input/{DATASET_FOLDER_NAME}'
final_data_dir = os.path.join(base_input_dir, 'Data_Final (Prev)') if os.path.exists(os.path.join
(base_input_dir, 'Data_Final (Prev)')) else base_input_dir

output_dir = '/kaggle/working/'
models_dir = os.path.join(output_dir, 'models')
results_dir = os.path.join(output_dir, 'results_final')
os.makedirs(models_dir, exist_ok=True)
os.makedirs(results_dir, exist_ok=True)

print(f"Data source: {final_data_dir}")
print(f"Models will be saved to: {models_dir}")
print(f"Results will be saved to: {results_dir}")
```

```
Data source: /kaggle/input/chest-x-ray-5-class-classification-dataset/Data_Final (Prev)
Models will be saved to: /kaggle/working/models
Results will be saved to: /kaggle/working/results_final
```

Figure 5.3 (e) : Define Paths

This section sets up the file directory structure for the Kaggle environment by defining paths for the dataset input location, model storage, and results output. The code automatically detects the chest X-ray dataset location and creates necessary output folders for saving trained models and evaluation results. The path configuration ensures organized file management throughout the training workflow, with confirmation messages displaying the established directory locations.

5.3.2 Data Understanding



```
TRAIN SET:
  NORMAL: 1266 images
  PNEUMONIA: 3418 images
  COVID19: 1650 images
  TUBERCULOSIS: 1745 images
  LUNG_OPACITY: 1050 images
  Total: 9129 images

VALIDATION SET:
  NORMAL: 269 images
  PNEUMONIA: 983 images
  COVID19: 359 images
  TUBERCULOSIS: 374 images
  LUNG_OPACITY: 225 images
  Total: 2210 images

TEST SET:
  NORMAL: 317 images
  PNEUMONIA: 855 images
  COVID19: 371 images
  TUBERCULOSIS: 375 images
  LUNG_OPACITY: 225 images
  Total: 2143 images
```

Figure 5.3.2 (a) :Images in Each Folder

The Figure 5.3 (e) shows the number of images in each of the five medical classes (**NORMAL**, **PNEUMONIA**, **COVID19**, **TUBERCULOSIS**, **LUNG_OPACITY**) across the train, validation, and test datasets. This provides essential dataset statistics to understand class distribution and identify potential data imbalance issues before model training. The dataset comprises chest X-ray images distributed across five pathological classes: Normal, Pneumonia, COVID-19, Tuberculosis, and Lung Opacity, with a total of 13,490 images split into train (9,129 images), validation (2,218 images), and test (2,143 images) sets. The distribution reveals class imbalance, with Pneumonia having the highest number of images (3,418 in train set) and classes like Normal and Lung Opacity having comparatively fewer samples.

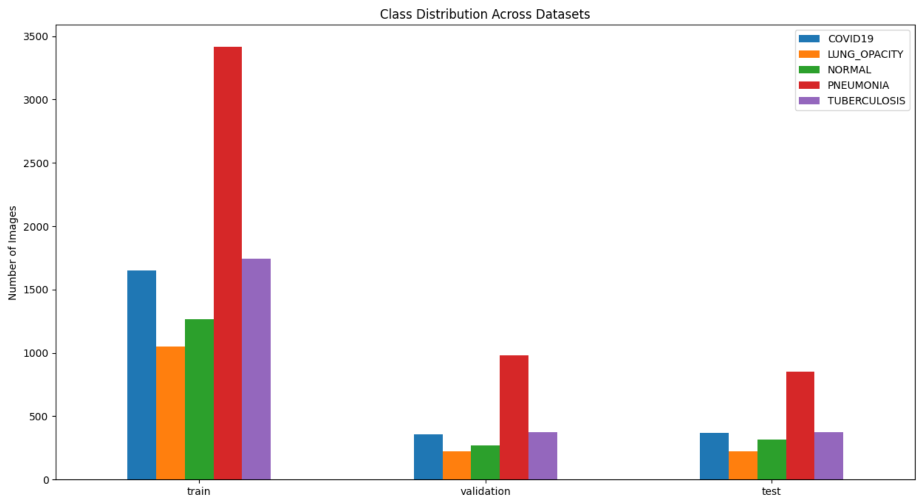


Figure 5.3.2 (b) : Class Distribution Analysis

This section analyses and visualizes the distribution of images across all five medical classes in the train, validation, and test datasets using a pandas DataFrame and bar chart. The analysis reveals potential class imbalance issues that may require addressing during model training through techniques like class weighting or data augmentation strategies.

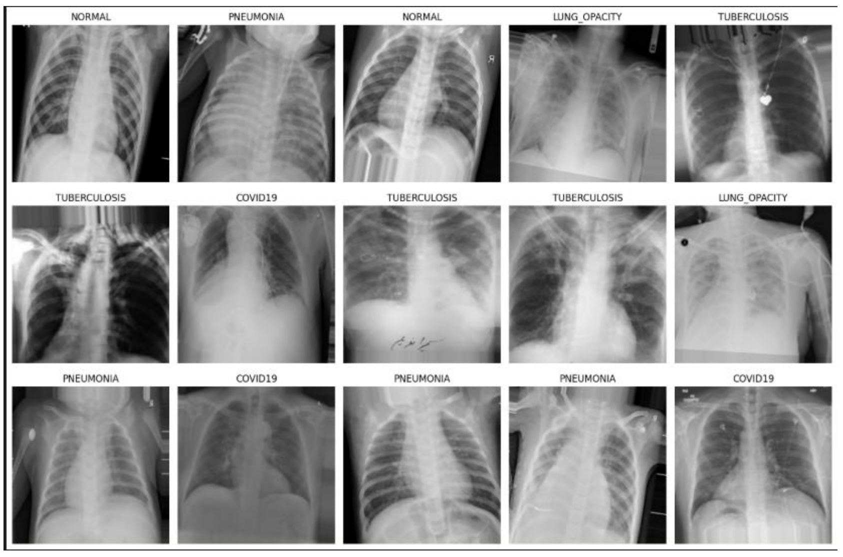
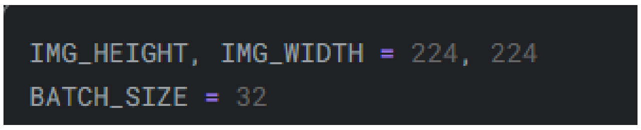


Figure 5.3.2 (c) : Visualize Sample Images

This section creates a visual display function that shows a batch of 15 sample chest X-ray images arranged in a 3×5 grid with their corresponding class labels. The function extracts images from the training data generator and displays them using **matplotlib**, providing a visual inspection of the dataset quality and ensuring proper data loading and labeling before model training begins.

5.3.3 Data Preprocessing

Image and Batch Configuration

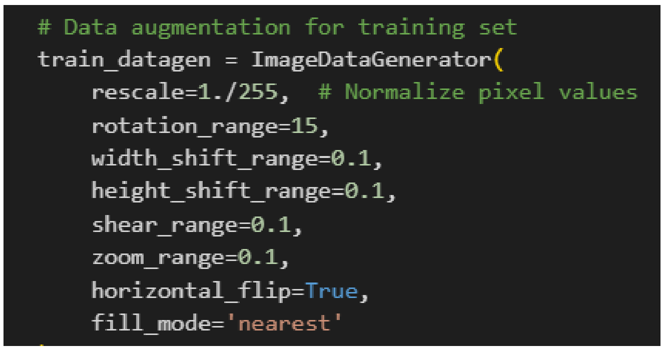


```
IMG_HEIGHT, IMG_WIDTH = 224, 224
BATCH_SIZE = 32
```

Figure 5.3.3 (a) : Define Image Dimension and Batch Size

Figure 5.3.3 (a) indicates the standard image dimensions (224×224 pixels) required for DenseNet121 input and set the batch size to 32 images per training iteration. These parameters ensure compatibility with the pre-trained model architecture and establish optimal memory usage during the training process.

Data Augmentation

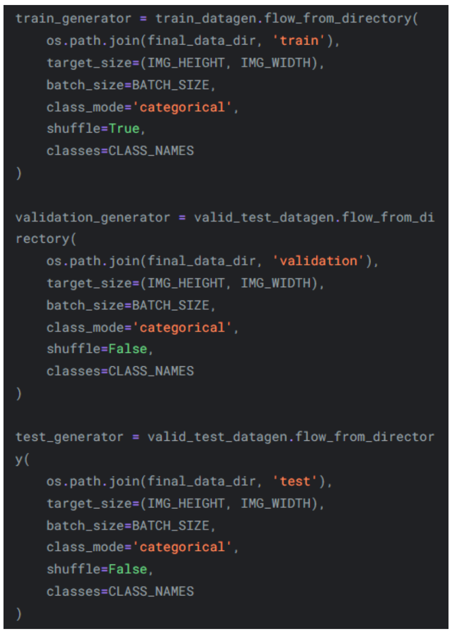


```
# Data augmentation for training set
train_datagen = ImageDataGenerator(
    rescale=1./255,  # Normalize pixel values
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
```

Figure 5.3.3 (b) : Data Augmentation

The Figure 5.3.3 indicates the section of creating an **ImageDataGenerator** for the training dataset to perform essential preprocessing and data augmentation. The **rescale** parameter normalizes pixel values from a 0-255 range to a 0-1 range, which is required by neural networks. Augmentation techniques include **rotation** (±10 degrees), **horizontal** and **vertical shifts** (10% and 5% respectively), **shearing** and **zooming** transformations (10% each), and **horizontal flipping**. These augmentation parameters are calibrated for chest X-ray images to artificially expand the training dataset while maintaining anatomical accuracy, which helps the model generalize better and reduces overfitting without compromising the clinical validity of the radiological features.

Data Generator Creation

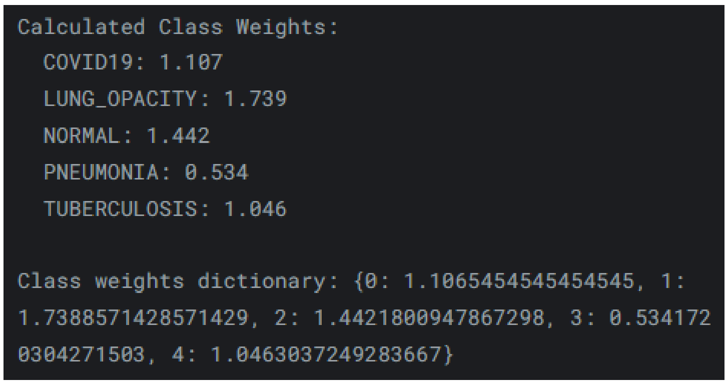


```
train_generator = train_datagen.flow_from_directory(
    os.path.join(final_data_dir, 'train'),
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=True,
    classes=CLASS_NAMES
)

validation_generator = valid_test_datagen.flow_from_di
rectory(
    os.path.join(final_data_dir, 'validation'),
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False,
    classes=CLASS_NAMES
)

test_generator = valid_test_datagen.flow_from_director
y(
    os.path.join(final_data_dir, 'test'),
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False,
    classes=CLASS_NAMES
)
```

Figure 5.3.3 (c) : Data Generator Creation

This code creates separate data generators for training, validation, and test datasets with appropriate configurations for each phase. The validation and test generators use only pixel normalization without augmentation to ensure consistent evaluation, while all generators are configured with categorical labels, a 224×224 target size, and a batch size of 32. The training generator includes shuffling for randomized learning, whereas validation and test generators maintain a fixed order for reproducible evaluation results.

Class Imbalance Handling

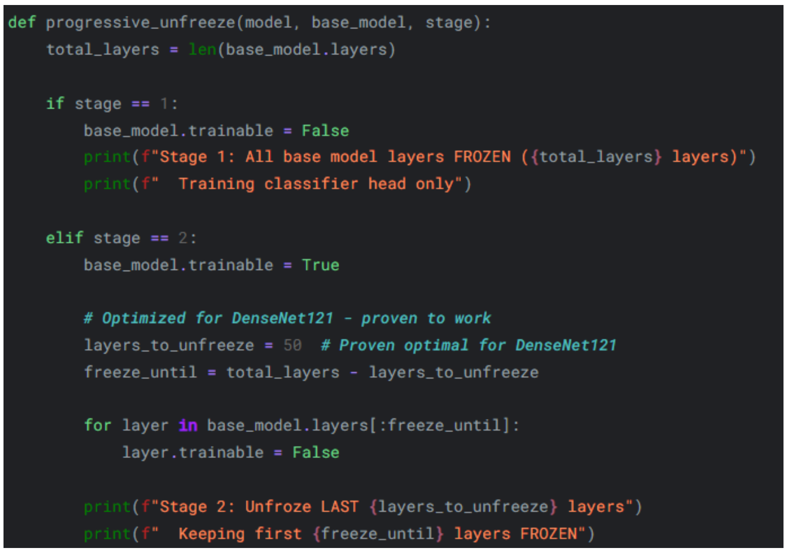


```
Calculated Class Weights:
  COVID19: 1.107
  LUNG_OPACITY: 1.739
  NORMAL: 1.442
  PNEUMONIA: 0.534
  TUBERCULOSIS: 1.046

Class weights dictionary: {0: 1.1065454545454545, 1:
1.7388571428571429, 2: 1.4421800947867298, 3: 0.534172
0304271503, 4: 1.0463037249283667}
```

Figure 5.3.4 (d) : Class Weight Calculation

This section calculates balanced class weights to address the unequal distribution of medical conditions in the chest X-ray dataset. The function automatically assigns higher weights to underrepresented diseases and lower weights to common conditions, ensuring the model learns equally from all five medical classes during training. These weights prevent the model from being biased toward frequent conditions like normal cases while improving the detection of rare diseases like tuberculosis.

5.3.4 Modelling

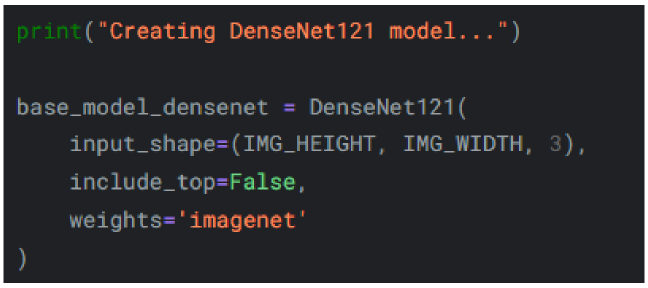


```
def progressive_unfreeze(model, base_model, stage):
    total_layers = len(base_model.layers)

    if stage == 1:
        base_model.trainable = False
        print(f"Stage 1: All base model layers FROZEN ({total_layers} layers)")
        print(f"  Training classifier head only")

    elif stage == 2:
        base_model.trainable = True

        # Optimized for DenseNet121 - proven to work
        layers_to_unfreeze = 50  # Proven optimal for DenseNet121
        freeze_until = total_layers - layers_to_unfreeze

        for layer in base_model.layers[:freeze_until]:
            layer.trainable = False

        print(f"Stage 2: Unfroze LAST {layers_to_unfreeze} layers")
        print(f"  Keeping first {freeze_until} layers FROZEN")
```

Figure 5.3.4 (a) : Progressive Unfreezing Strategy

This section implements a two-stage training approach for DenseNet121 optimization in medical image classification. Stage 1 trains only the custom classification head while keeping all base model layers frozen. Stage 2 then unfreezes the last 50 layers for fine-tuning while preserving earlier learned features. This progressive strategy prevents catastrophic forgetting of ImageNet features, enabling the model to adapt to chest X-ray-specific patterns for accurate medical diagnosis.

DenseNet121 Base Model Initialization

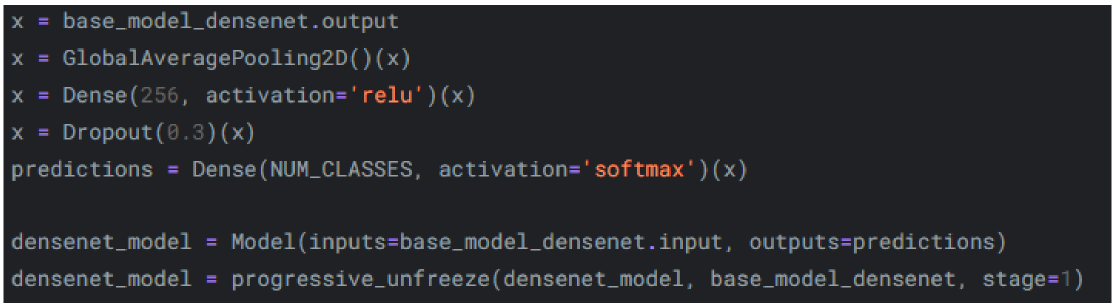


```
print("Creating DenseNet121 model...")

base_model_densenet = DenseNet121(
    input_shape=(IMG_HEIGHT, IMG_WIDTH, 3),
    include_top=False,
    weights='imagenet'
)
```

Figure 5.3.4 (b) : Model Development of DenseNet121

This code initializes the DenseNet121 pre-trained model as the feature extraction backbone for the chest X-ray classification task. The model is configured with an input shape of 224x224x3 pixels to match the preprocessed chest X-ray images, while include_top=False removes the original ImageNet classification head to allow for custom medical classification layers. The weights='imagenet' parameter loads pre-trained weights from ImageNet, providing a strong foundation of learned visual features that will be adapted for medical image analysis through transfer learning.

Custom Classification Head

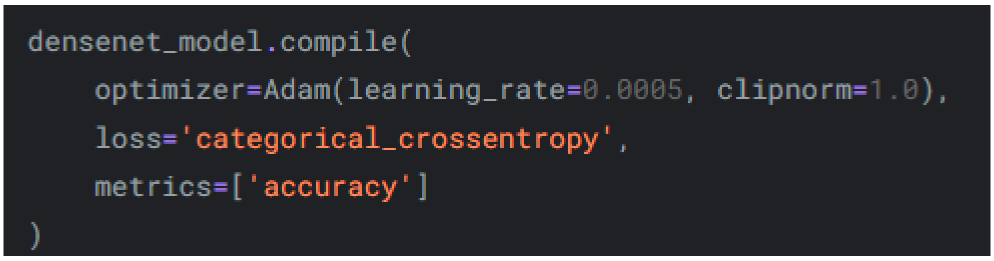


```
x = base_model_densenet.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.3)(x)
predictions = Dense(NUM_CLASSES, activation='softmax')(x)

densenet_model = Model(inputs=base_model_densenet.input, outputs=predictions)
densenet_model = progressive_unfreeze(densenet_model, base_model_densenet, stage=1)
```

Figure 5.3.4 (c) : Custom Classification Head

This code builds a custom classification head on top of the DenseNet121 base model for medical image classification. The architecture applies GlobalAveragePooling2D to convert the feature maps into a 1D vector, followed by a Dense layer with 256 neurons and ReLU activation for feature learning. A Dropout layer with a 0.3 rate is then used for regularization to prevent overfitting. Finally, a softmax output layer with 5 neurons corresponds to the medical classes (COVID19, LUNG_OPACITY, NORMAL, PNEUMONIA, TUBERCULOSIS). The complete model is assembled by connecting the base model input to the custom prediction output, and a progressive unfreezing strategy is applied starting with Stage 1 (frozen base model).

Model Compilation



```
densenet_model.compile(
    optimizer=Adam(learning_rate=0.0005, clipnorm=1.0),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

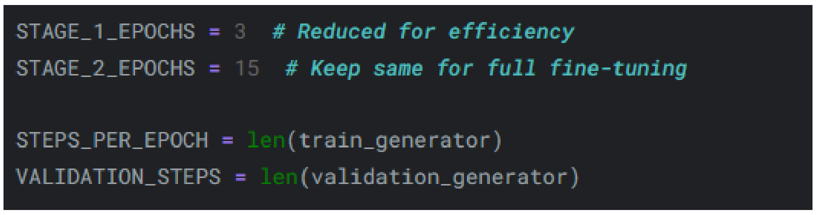
Figure 5.3.4 (d): Model Compilation

Adam optimizer: Learning rate 0.0005 with gradient clipping (clipnorm=1.0) for stable training.

Categorical crossentropy: Loss function for 5-class medical classification.

Accuracy metric: Primary evaluation measure for training monitoring.

Training Configuration Parameters



```
STAGE_1_EPOCHS = 3  # Reduced for efficiency
STAGE_2_EPOCHS = 15  # Keep same for full fine-tuning

STEPS_PER_EPOCH = len(train_generator)
VALIDATION_STEPS = len(validation_generator)
```

Figure 5.3.4 (e):Training Configuration

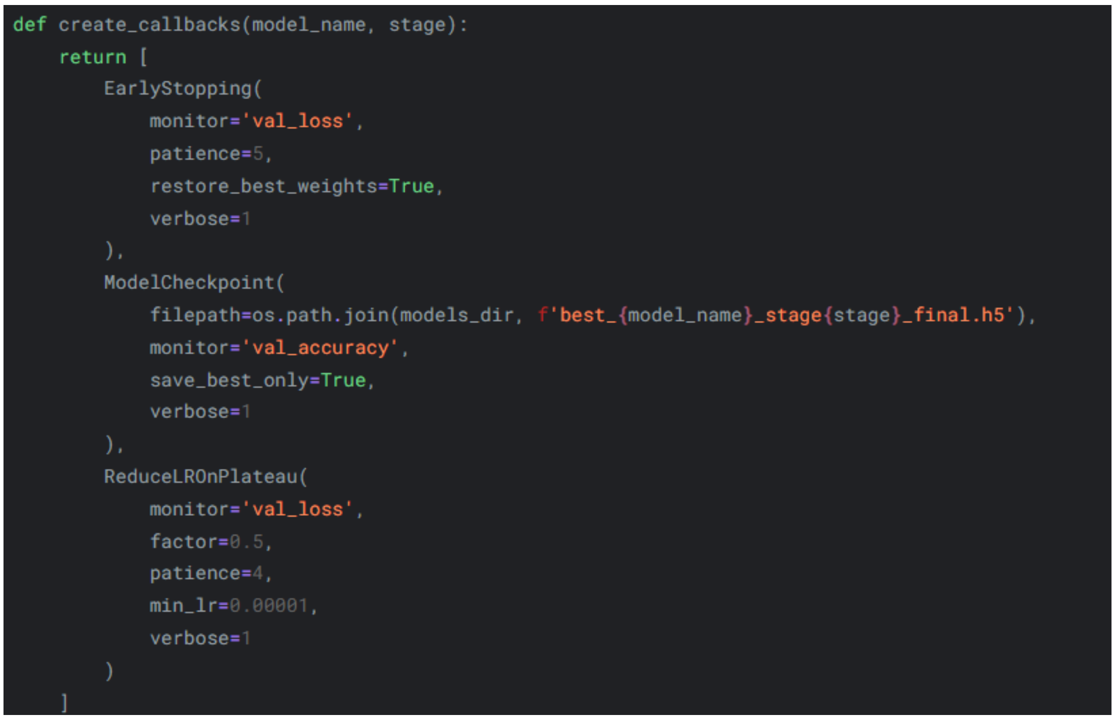
Stage 1: Training for 3 epochs to efficiently train the classifier head with a frozen base model.

Stage 2: Fine-tuning for 15 epochs with unfrozen layers for comprehensive model optimization.

Steps Calculation: The number of steps per epoch is calculated using the length of the data generators, ensuring the entire dataset is covered in each epoch.

Training Efficiency: This two-stage approach balances computational efficiency with effective model optimization.

Training Callbacks Configuration

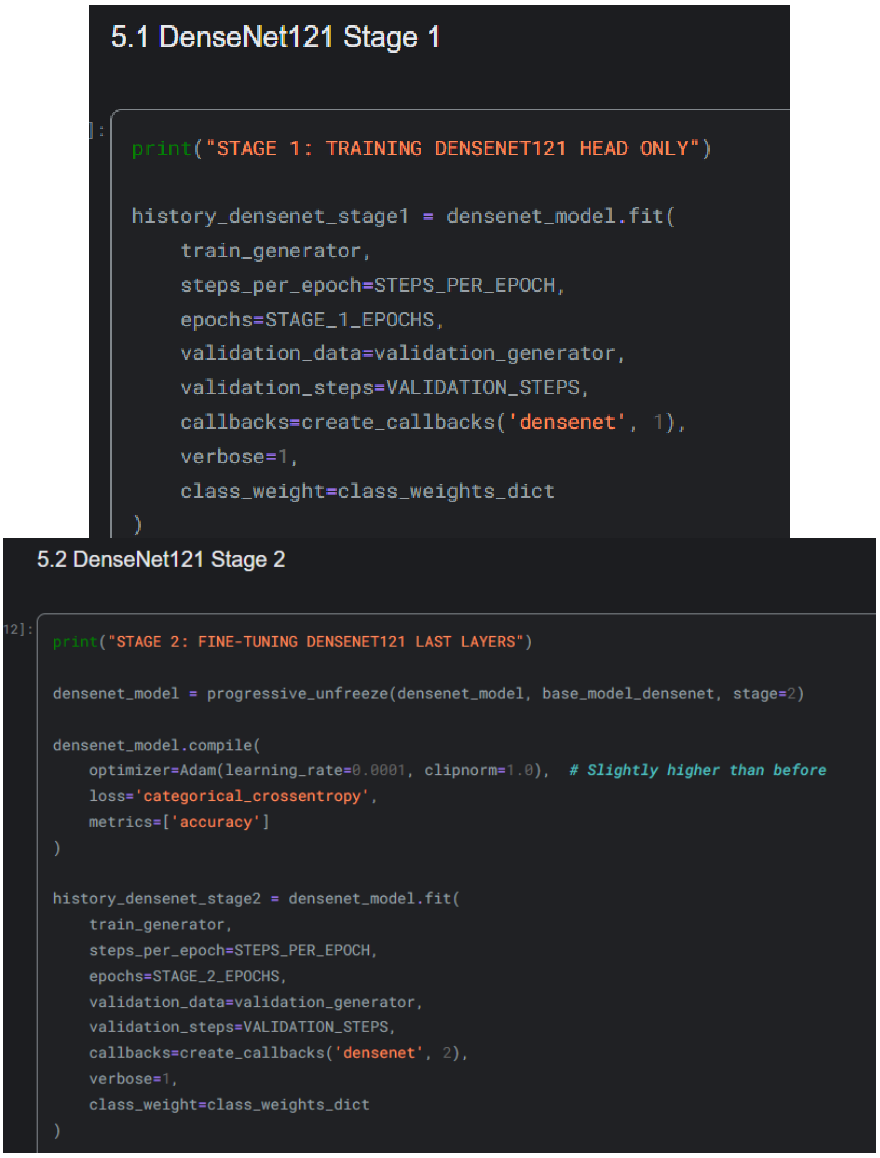


```
def create_callbacks(model_name, stage):
    return [
        EarlyStopping(
            monitor='val_loss',
            patience=5,
            restore_best_weights=True,
            verbose=1
        ),
        ModelCheckpoint(
            filepath=os.path.join(models_dir, f'best_{model_name}_stage{stage}_final.h5'),
            monitor='val_accuracy',
            save_best_only=True,
            verbose=1
        ),
        ReduceLROnPlateau(
            monitor='val_loss',
            factor=0.5,
            patience=4,
            min_lr=0.00001,
            verbose=1
        )
    ]
```

Figure 5.3.4 (f): Callback for DenseNet121 Training

The function shown in Figure 5.3.4 (f) creates essential training callbacks to monitor and optimize the model training process for both progressive unfreezing stages. The **EarlyStopping** callback prevents overfitting by halting training when the validation loss does not improve for 5 epochs and restores the best model weights. The **ModelCheckpoint** callback saves the best-performing model based on validation accuracy for each stage. The **ReduceLROnPlateau** callback dynamically reduces the learning rate by half when the validation loss plateaus for 4 epochs, which helps to fine-tune convergence. These callbacks work together to ensure robust training by preventing overfitting, preserving the best models, and adapting learning rates for optimal performance.

DenseNet121 Training



5.1 DenseNet121 Stage 1

```
print("STAGE 1: TRAINING DENSENET121 HEAD ONLY")

history_densenet_stage1 = densenet_model.fit(
    train_generator,
    steps_per_epoch=STEPS_PER_EPOCH,
    epochs=STAGE_1_EPOCHS,
    validation_data=validation_generator,
    validation_steps=VALIDATION_STEPS,
    callbacks=create_callbacks('densenet', 1),
    verbose=1,
    class_weight=class_weights_dict
)
```

5.2 DenseNet121 Stage 2

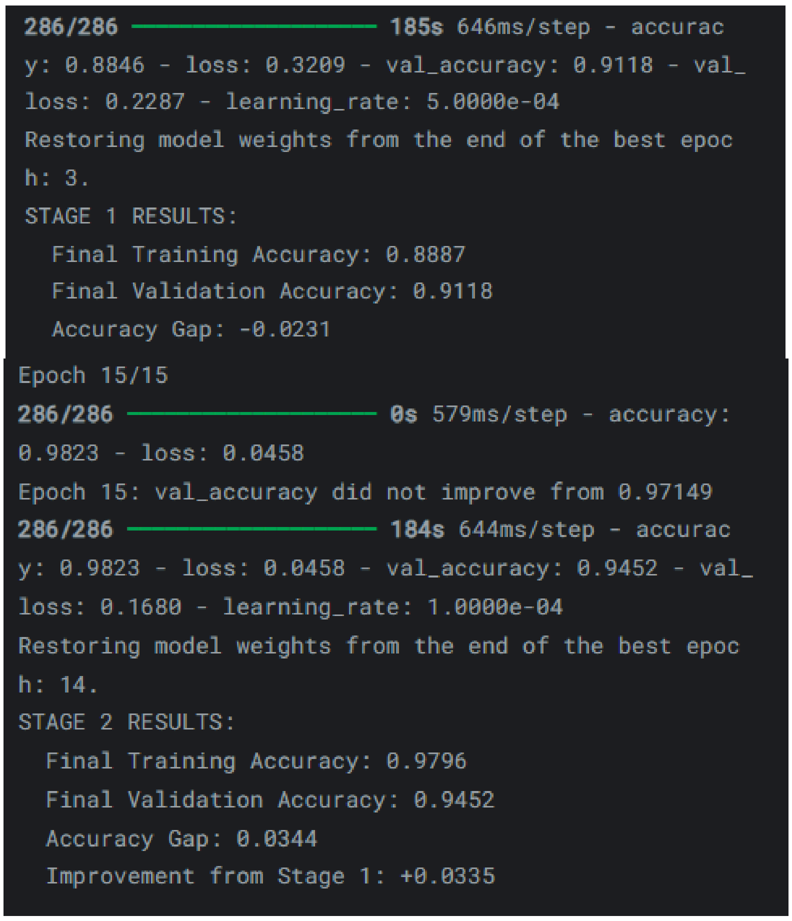
```
print("STAGE 2: FINE-TUNING DENSENET121 LAST LAYERS")

densenet_model = progressive_unfreeze(densenet_model, base_model_densenet, stage=2)

densenet_model.compile(
    optimizer=Adam(learning_rate=0.0001, clipnorm=1.0),  # Slightly higher than before
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

history_densenet_stage2 = densenet_model.fit(
    train_generator,
    steps_per_epoch=STEPS_PER_EPOCH,
    epochs=STAGE_2_EPOCHS,
    validation_data=validation_generator,
    validation_steps=VALIDATION_STEPS,
    callbacks=create_callbacks('densenet', 2),
    verbose=1,
    class_weight=class_weights_dict
)
```

Figure 5.3.4 (g): Training DenseNet121

Stage 1 (Head Training). This phase executes the first stage of **progressive unfreezing** by training only the classification head while keeping all **DenseNet121** base layers frozen. The model is trained for 3 epochs using the training generator with **class weights** to handle imbalanced medical data. Validation performance is monitored through callbacks like **EarlyStopping** and **ModelCheckpoint**. The results—including training accuracy, validation accuracy, and the accuracy gap—are recorded to assess the initial classifier performance before fine-tuning.

Stage 2 (Fine-tuning). This phase implements fine-tuning by unfreezing the last 50 layers of **DenseNet121** and recompiling the model with a reduced learning rate of 0.0001 to prevent **catastrophic forgetting**. The model undergoes 15 epochs of training with the same data generators and callbacks. This allows the unfrozen layers to adapt to chest X-ray-specific features while maintaining previously learned representations. The results track the improvement in validation accuracy compared to Stage 1, demonstrating the effectiveness of the progressive unfreezing approach for medical image classification.

Training Results Summary

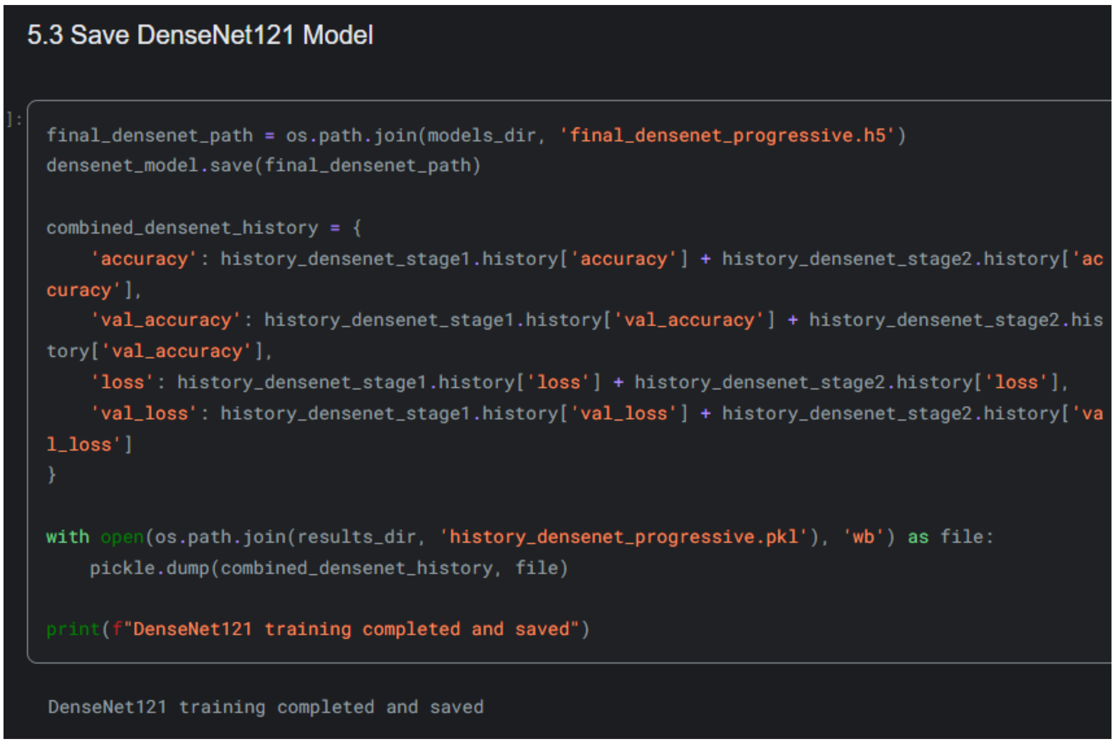


```
286/286 ━━━━━━━━━━━━━━━━━━━━ 185s 646ms/step - accurac
y: 0.8846 - loss: 0.3209 - val_accuracy: 0.9118 - val_
loss: 0.2287 - learning_rate: 5.0000e-04
Restoring model weights from the end of the best epoc
h: 3.
STAGE 1 RESULTS:
  Final Training Accuracy: 0.8887
  Final Validation Accuracy: 0.9118
  Accuracy Gap: -0.0231
Epoch 15/15
286/286 ━━━━━━━━━━━━━━━━━━━━ 0s 579ms/step - accuracy:
0.9823 - loss: 0.0458
Epoch 15: val_accuracy did not improve from 0.97149
286/286 ━━━━━━━━━━━━━━━━━━━━ 184s 644ms/step - accurac
y: 0.9823 - loss: 0.0458 - val_accuracy: 0.9452 - val_
loss: 0.1680 - learning_rate: 1.0000e-04
Restoring model weights from the end of the best epoc
h: 14.
STAGE 2 RESULTS:
  Final Training Accuracy: 0.9796
  Final Validation Accuracy: 0.9452
  Accuracy Gap: 0.0344
  Improvement from Stage 1: +0.0335
```

Figure 5.3.4 (h): Training Results Summary

Stage 1 of the training achieved strong initial performance with 88.87% training accuracy and 91.18% validation accuracy, demonstrating good generalization as validation accuracy exceeded training accuracy by 2.31%. **Stage 2** further enhanced the model's performance through progressive unfreezing, reaching 97.96% training accuracy and 94.52% validation accuracy, a 3.35% improvement in validation performance that confirms the effectiveness of the fine-tuning approach for medical image classification.

Saving Model



5.3 Save DenseNet121 Model

```
final_densenet_path = os.path.join(models_dir, 'final_densenet_progressive.h5')
densenet_model.save(final_densenet_path)

combined_densenet_history = {
    'accuracy': history_densenet_stage1.history['accuracy'] + history_densenet_stage2.history['ac
curacy'],
    'val_accuracy': history_densenet_stage1.history['val_accuracy'] + history_densenet_stage2.his
tory['val_accuracy'],
    'loss': history_densenet_stage1.history['loss'] + history_densenet_stage2.history['loss'],
    'val_loss': history_densenet_stage1.history['val_loss'] + history_densenet_stage2.history['va
l_loss']
}

with open(os.path.join(results_dir, 'history_densenet_progressive.pkl'), 'wb') as file:
    pickle.dump(combined_densenet_history, file)

print(f"DenseNet121 training completed and saved")
```

```
DenseNet121 training completed and saved
```

Figure 5.3.4 (i) : Save DenseNet121 Model

Figure 5.3.4 (i) shows the step of saving the trained DenseNet121 model and consolidates the training history from both stages into a single, comprehensive record. The final model is saved as **final_densenet_progressive.h5**. The combined training metrics, including accuracy and loss from Stage 1 and Stage 2, are serialized using pickle for future analysis and visualization. This ensures that the complete training process and optimal model weights are preserved for deployment and evaluation.

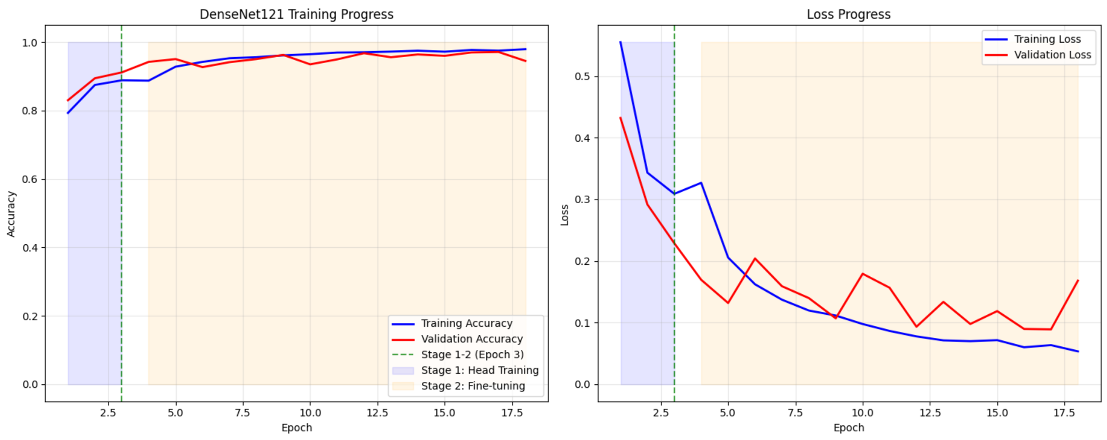
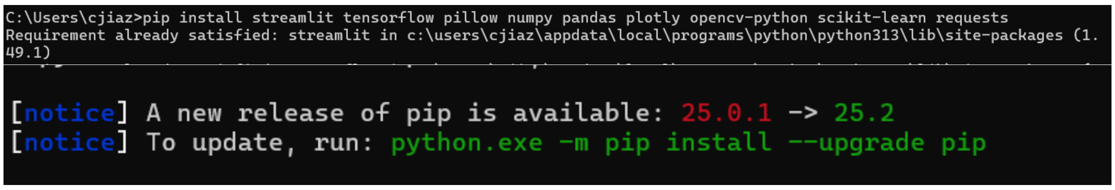


Figure 5.3.4 (j) : Training Visualization

The visualization above reveals the model's performance progression across 18 epochs during the two-stage progressive unfreezing approach. The figure displays training and validation accuracy curves, with distinct color-coded regions highlighting the transition from head training to fine-tuning phases. The loss progress chart illustrates training and validation loss curves throughout the process. The close alignment between training and validation curves indicates excellent model generalization without significant overfitting, suggesting the progressive unfreezing strategy successfully balanced learning capacity with regularization for optimal performance.

5.4 System Setting and Operation

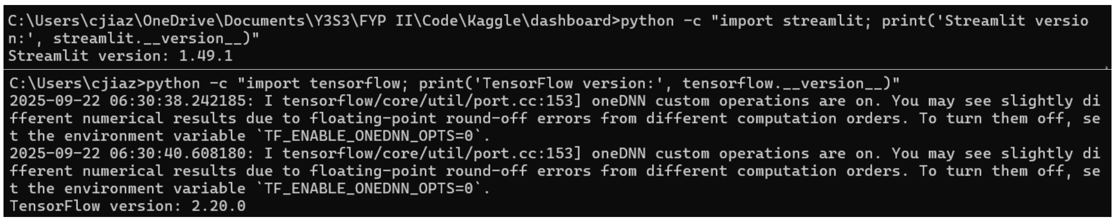
The chest X-ray AI diagnostic system was implemented as an interactive web application using the **Streamlit** framework, developed and deployed on a local Windows environment for comprehensive testing and validation. The dashboard integrates the trained DenseNet121 model with a user-friendly interface that provides real-time chest X-ray analysis, AI interpretability through Grad-CAM visualization, and supplementary health monitoring features including weather health alerts and hospital finder functionality. The system architecture follows a modular design with separate components for image processing, model prediction, visualization, and external API integration, ensuring maintainable and scalable code structure. The Streamlit framework enables rapid prototyping and deployment while providing responsive web components that deliver professional medical-grade user experience with minimal development overhead. The local deployment environment allows for immediate testing, model iteration, and user interface refinement, establishing a foundation for future cloud deployment and broader accessibility.



```
C:\Users\cjiaz>pip install streamlit tensorflow pillow numpy pandas plotly opencv-python scikit-learn requests
Requirement already satisfied: streamlit in c:\users\cjiaz\appdata\local\programs\python\python313\lib\site-packages (1.
49.1)

[notice] A new release of pip is available: 25.0.1 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Figure 5.4 (a) : Package Installation

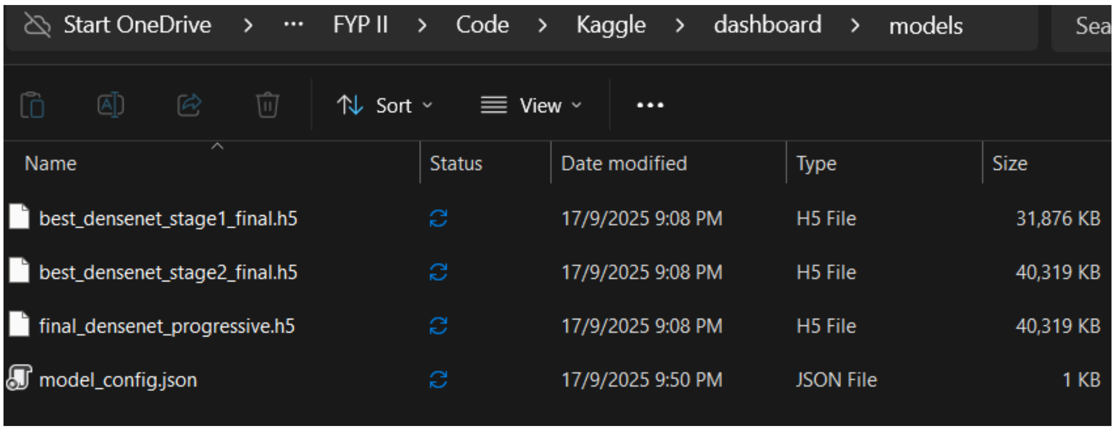
This command installs all required Python libraries for the dashboard. Streamlit creates the web interface, TensorFlow loads the AI model, Pillow handles image processing, and other libraries support data visualization and external API integration.



```
C:\Users\cjiaz\OneDrive\Documents\Y3S3\FYP II\Code\Kaggle\dashboard>python -c "import streamlit; print('Streamlit versio
n:', streamlit.__version__)"
Streamlit version: 1.49.1

C:\Users\cjiaz>python -c "import tensorflow; print('TensorFlow version:', tensorflow.__version__)"
2025-09-22 06:30:38.242185: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly di
fferent numerical results due to floating-point round-off errors from different computation orders. To turn them off, se
t the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-09-22 06:30:40.608180: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly di
fferent numerical results due to floating-point round-off errors from different computation orders. To turn them off, se
t the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
TensorFlow version: 2.20.0
```

Figure 5.4 (b) : Installation Verification

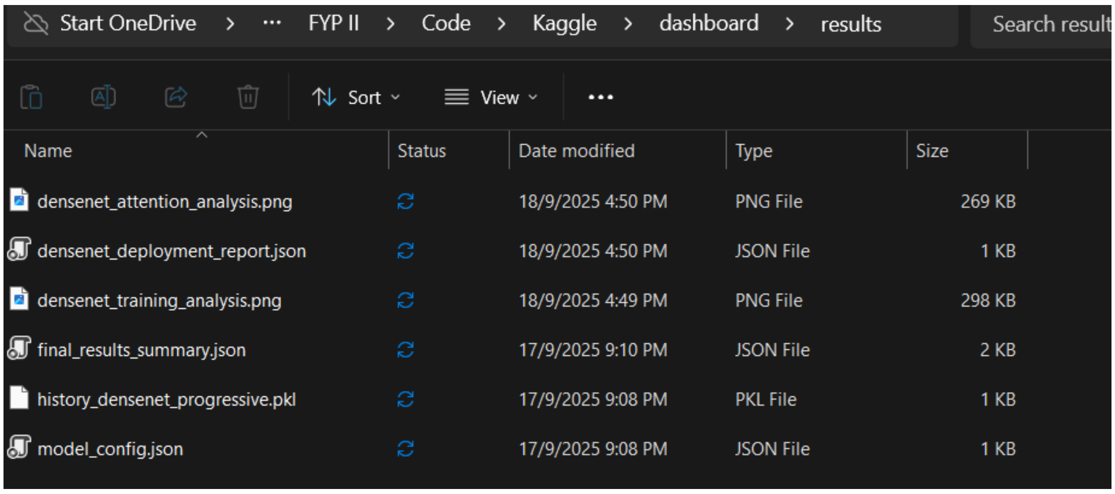
These commands verify that Streamlit and TensorFlow installed correctly by importing them and displaying their version numbers. This ensures the core components needed for the dashboard are working properly.



Start OneDrive > ··· FYP II > Code > Kaggle > dashboard > models

Name	Status	Date modified	Type	Size
best_densenet_stage1_final.h5		17/9/2025 9:08 PM	H5 File	31,876 KB
best_densenet_stage2_final.h5		17/9/2025 9:08 PM	H5 File	40,319 KB
final_densenet_progressive.h5		17/9/2025 9:08 PM	H5 File	40,319 KB
model_config.json		17/9/2025 9:50 PM	JSON File	1 KB

Figure 5.4 (c) : Models File Transfer


Start OneDrive > ··· FYP II > Code > Kaggle > dashboard > results

Name	Status	Date modified	Type	Size
densenet_attention_analysis.png		18/9/2025 4:50 PM	PNG File	269 KB
densenet_deployment_report.json		18/9/2025 4:50 PM	JSON File	1 KB
densenet_training_analysis.png		18/9/2025 4:49 PM	PNG File	298 KB
final_results_summary.json		17/9/2025 9:10 PM	JSON File	2 KB
history_densenet_progressive.pkl		17/9/2025 9:08 PM	PKL File	1 KB
model_config.json		17/9/2025 9:08 PM	JSON File	1 KB

Figure 5.4 (d) : Results File Transfer

From Figure 5.4 (c) and Figure 5.4 (d) shows that the step of transferring the trained AI model file and configuration settings from Kaggle to the local dashboard project. The .h5 file contains the trained DenseNet121 model weights, while the JSON file contains model specifications and class names.

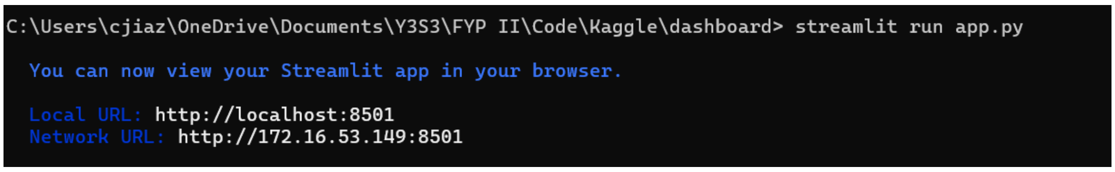
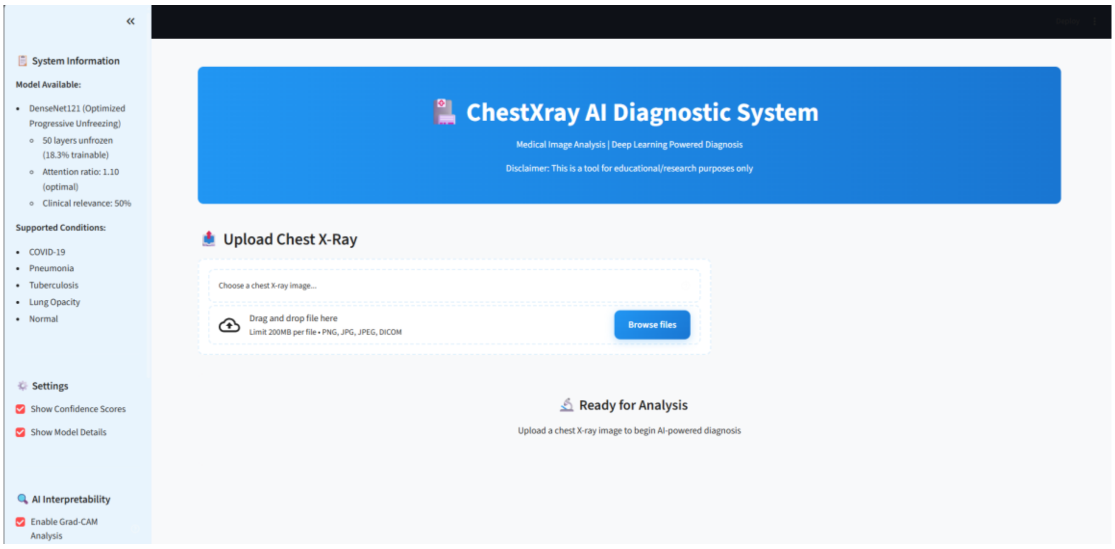

```
C:\Users\cjiaz\OneDrive\Documents\Y3S3\FYP II\Code\Kaggle\dashboard> streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://172.16.53.149:8501
```

Figure 5.4 (e) : Dashboard Launch

This command starts the Streamlit web server and launches the dashboard application. It opens a local web server at http://localhost:8501 where users can access the chest X-ray diagnostic interface through their web browser.

Homepage

Image Upload Process


System Information

Model Available:

- DenseNet121 (Optimized Progressive Unfreezing)
  - 50 layers unfrozen (18.3% trainable)
  - Attention ratio: 1.10 (optimal)
  - Clinical relevance: 50%

Supported Conditions:

- COVID-19
- Pneumonia
- Tuberculosis
- Lung Opacity
- Normal

Settings

- [✓] Show Confidence Scores
- [✓] Show Model Details

AI Interpretability

- [✓] Enable Grad-CAM Analysis

ChestXray AI Diagnostic System

Medical Image Analysis | Deep Learning Powered Diagnosis

Disclaimer: This is a tool for educational/research purposes only

Upload Chest X-Ray

Choose a chest X-ray image...

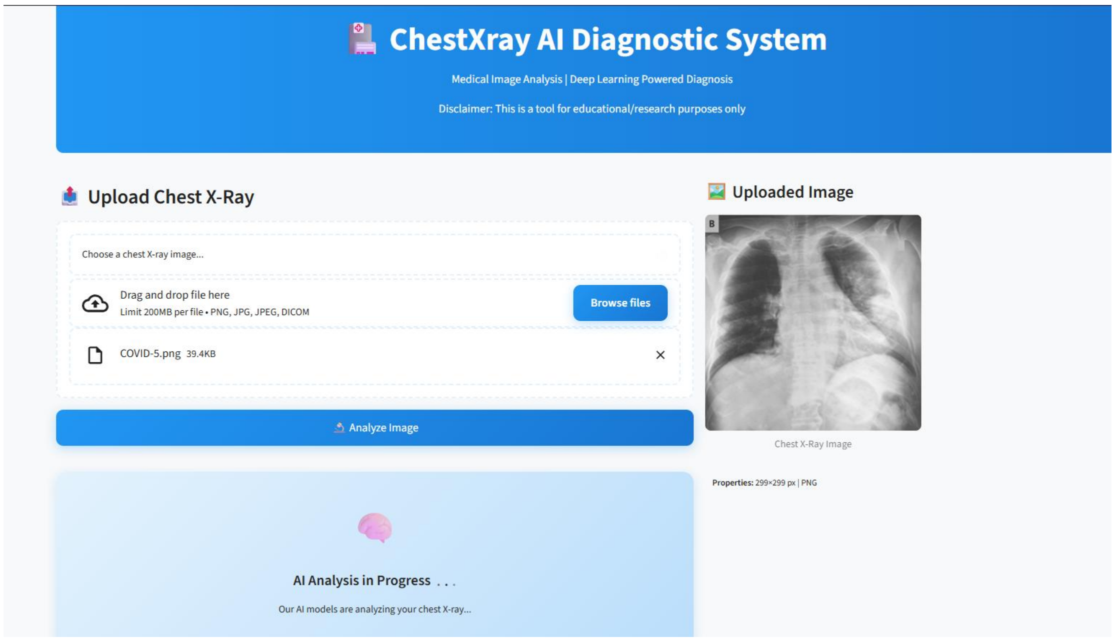
Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG, DICOM

Browse files

Ready for Analysis

Upload a chest X-ray image to begin AI-powered diagnosis

Figure 5.4 (f) : Image Upload


ChestXray AI Diagnostic System

Medical Image Analysis | Deep Learning Powered Diagnosis

Disclaimer: This is a tool for educational/research purposes only

Upload Chest X-Ray

Choose a chest X-ray image...

Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG, DICOM

Browse files

COVID-5.png 39.4KB

Analyze Image

AI Analysis in Progress . . .

Our AI models are analyzing your chest X-ray...

Uploaded Image

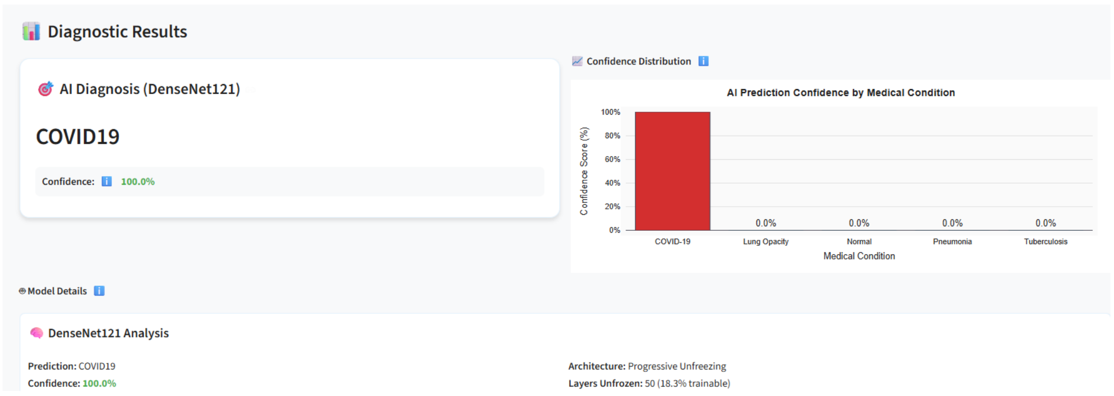
Chest X-Ray Image

Properties: 299×299 px | PNG

Figure 5.4 (g) : Image under Analysis Process

The Figure 5.4 (f) and Figure 5.4 (g) shows the user interaction begins with accessing the main dashboard interface where a prominent file upload component allows users to select and upload chest X-ray images medical images. Users can easily select chest X-ray images in common formats like PNG, JPG, or JPEG. The system quickly validates the file, displays a preview, and provides basic image details. After selecting an image, users can click an "Analyze X-ray" button to initiate AI processing. The system then preprocesses the image, resizing and normalizing it before running the DenseNet121 model to generate medical condition predictions. This streamlined workflow allows users to obtain diagnostic insights quickly and effortlessly, without requiring technical expertise.

AI Prediction Results



Diagnostic Results

AI Diagnosis (DenseNet121)

COVID19

Confidence: 100.0%

Model Details

DenseNet121 Analysis

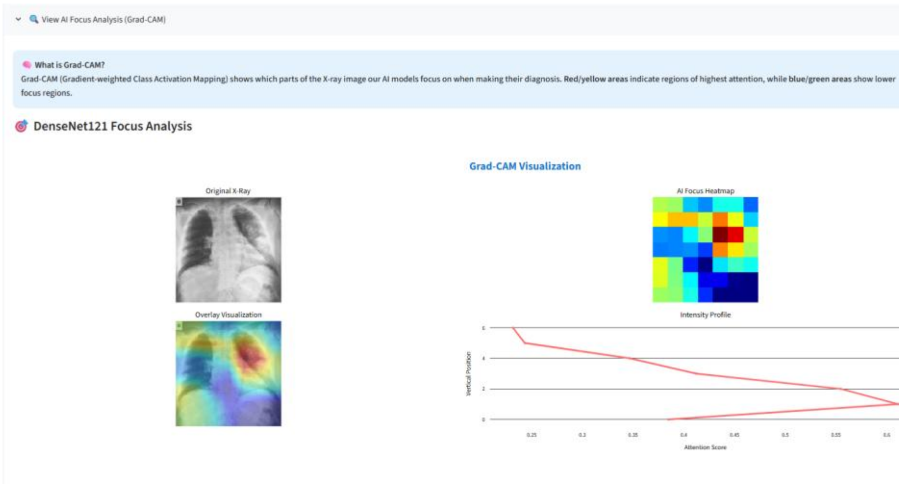
Prediction: COVID19
Confidence: 100.0%

Architecture: Progressive Unfreezing
Layers Unfrozen: 50 (18.3% trainable)

Figure 5.4 (h) : Diagnostic Results Shown

The diagnostic results section displays the AI's primary prediction among the five chest conditions (COVID-19, Normal, Pneumonia, Tuberculosis, Lung Opacity) with a clear confidence score shown as both percentage and color-coded indicators. Users can view the complete probability distribution for all conditions through an interactive chart, enabling assessment of alternative diagnoses and their likelihood. The system includes safety warnings when confidence levels fall below clinical thresholds (75-80%), helping users understand prediction reliability and determine if additional medical evaluation is needed.

Viewing AI Focus Analysis (Grad-CAM)



View AI Focus Analysis (Grad-CAM)

What is Grad-CAM?
Grad-CAM (Gradient-weighted Class Activation Mapping) shows which parts of the X-ray image our AI models focus on when making their diagnosis. **Red/yellow areas** indicate regions of highest attention, while **blue/green areas** show lower focus regions.

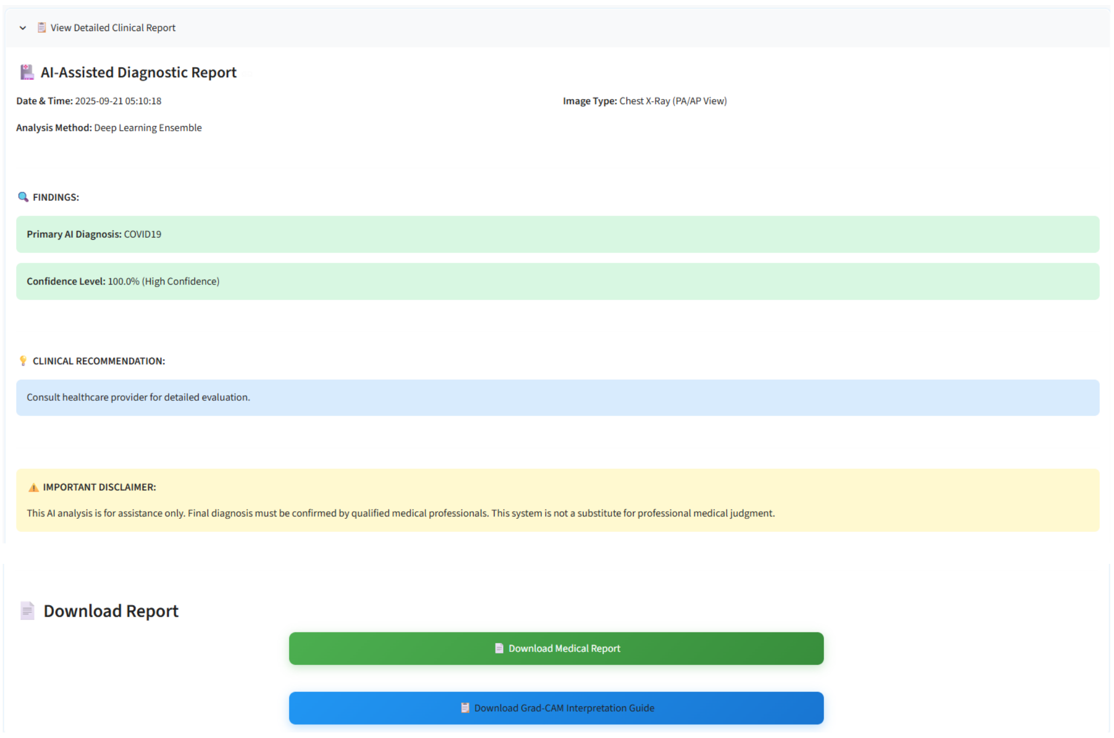
DenseNet121 Focus Analysis

Figure 5.4 (i) : Grad Cam Visualization

The AI Focus Analysis section provides users with visual interpretability through Grad-CAM (Gradient-weighted Class Activation Mapping) technology, enabling healthcare practitioners to understand how the AI model reaches its diagnostic conclusions. Upon selecting the "View AI Focus Analysis" option, users are presented with an overlay visualization that highlights the specific regions of the chest X-ray that influenced the AI's prediction, displayed as a color-coded heatmap superimposed on the original image.

The visualization uses a "jet" colormap where red areas indicate regions of highest attention (most influential for the diagnosis), yellow-orange areas show moderate influence, and blue areas represent minimal impact on the prediction. This interpretability feature builds trust between medical professionals and the AI system by making the "black box" nature of deep learning transparent, allowing clinicians to validate that the AI is examining appropriate radiological features rather than irrelevant artifacts, and supporting informed clinical decision-making by showing exactly where potential abnormalities were detected in the chest X-ray image.

Viewing Detailed Clinical Report



View Detailed Clinical Report

AI-Assisted Diagnostic Report

Date & Time: 2025-09-21 05:10:18 **Image Type:** Chest X-Ray (PA/AP View)

Analysis Method: Deep Learning Ensemble

FINDINGS:

Primary AI Diagnosis: COVID19

Confidence Level: 100.0% (High Confidence)

CLINICAL RECOMMENDATION:

Consult healthcare provider for detailed evaluation.

IMPORTANT DISCLAIMER:

This AI analysis is for assistance only. Final diagnosis must be confirmed by qualified medical professionals. This system is not a substitute for professional medical judgment.

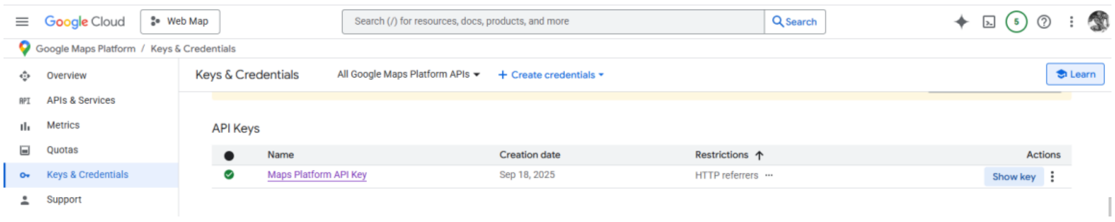
Download Report

Figure 5.4 (j) : Clinical Report

The Detailed Clinical Report feature generates a comprehensive, downloadable medical document that consolidates all AI analysis results into a professional clinical format suitable for medical records and healthcare documentation. The report includes patient information fields, the primary AI diagnosis with confidence levels, complete probability distributions across all five medical conditions, timestamp and system metadata, and clinical recommendations based on the predicted condition and confidence score.

Users can access this feature by clicking the "View Detailed Clinical Report" button, which presents a formatted document containing structured medical language, diagnostic summaries, and suggested follow-up actions that can be downloaded as a PDF or text file for integration into electronic health records, sharing with healthcare providers, or maintaining as part of the patient's medical documentation trail.

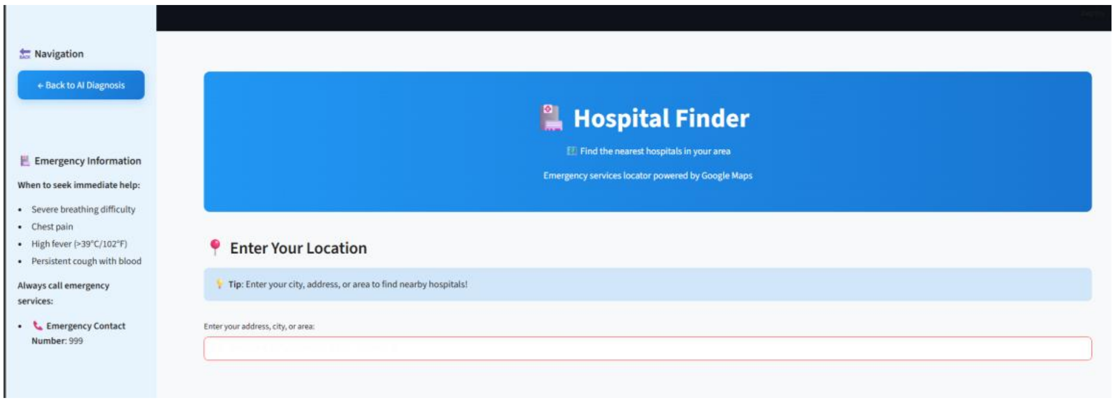
Hospital Finder Page



API Keys

●	Name	Creation date	Restrictions ↑	Actions
✓	Maps Platform API Key	Sep 18, 2025	HTTP referrers ···	Show key

Figure 5.4 (k) : Get Google Maps API Key


Navigation
← Back to AI Diagnosis
Hospital Finder
Find the nearest hospitals in your area
Emergency services locator powered by Google Maps
Enter Your Location

Emergency Information

When to seek immediate help:

- Severe breathing difficulty
- Chest pain
- High fever (>39°C/102°F)
- Persistent cough with blood

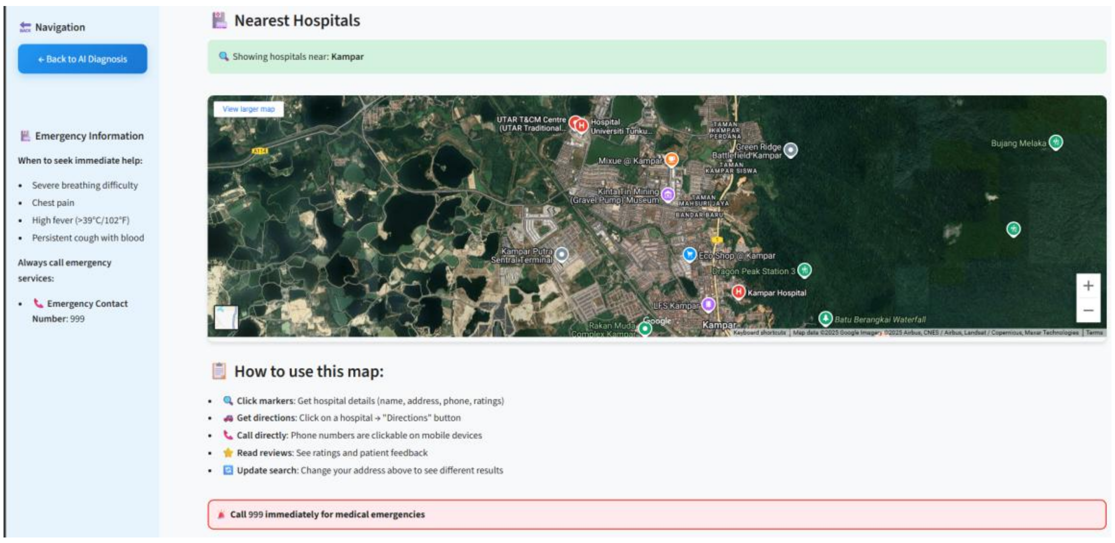
Always call emergency services:

- Emergency Contact Number: 999

Tip: Enter your city, address, or area to find nearby hospitals!

Enter your address, city, or area:

Figure 5.4 (l) : Enter location to Find Nearby Hospital


Navigation
← Back to AI Diagnosis
View larger map
UTAR T&CM Centre (UTAR Traditional...
Hospital Universiti Tunku...
Mixue @ Kampar
Green Ridge Battlefield Kampar
Kinta Tin Mining (Gravel Pump) Museum
Kampar Putra Sentral Terminal
Eco Shop @ Kampar
Dragon Peak Station 3
Kampar Hospital
Bujang Melaka
Batu Berangkai Waterfall
Kampar

Nearest Hospitals

🔍 Showing hospitals near: **Kampar**

Emergency Information

When to seek immediate help:

- Severe breathing difficulty
- Chest pain
- High fever (>39°C/102°F)
- Persistent cough with blood

Always call emergency services:

- 📞 **Emergency Contact Number:** 999

How to use this map:

- 🔍 **Click markers:** Get hospital details (name, address, phone, ratings)
- 🚑 **Get directions:** Click on a hospital → "Directions" button
- 📞 **Call directly:** Phone numbers are clickable on mobile devices
- ⭐ **Read reviews:** See ratings and patient feedback
- 🔄 **Update search:** Change your address above to see different results

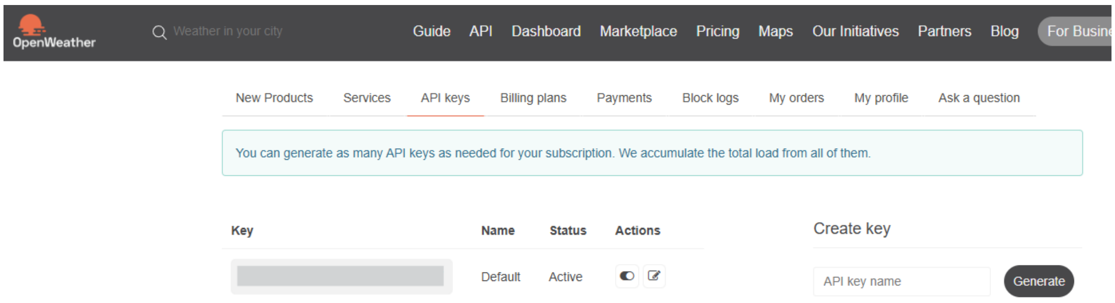
🚨 **Call 999 immediately for medical emergencies**

Figure 5.4 (m) : Google Maps showing the Nearest Hospital

From the Figure 5.4 (f) and Figure 5.4 (g) above, demonstrate the **Hospital Finder** feature provides users with immediate access to nearby medical facilities through integrated Google Maps technology, accessible via the "Check Nearest Hospital" button in the dashboard sidebar. Upon activation, users are presented with a dedicated page where they can enter their current location (city, address, or postal code) into a search interface, which then leverages **Google Maps Embed API** to display an interactive map showing nearby hospitals, clinics, and emergency medical centers.

The system automatically searches for healthcare facilities within the user's vicinity and presents results with essential information including hospital names, addresses, contact details, distance from the user's location, and direct navigation links. Users can interact with the map by zooming, panning, and clicking on hospital markers to view additional details, while the interface provides filtering options for different types of medical facilities (emergency rooms, general hospitals, specialized clinics). This feature is particularly valuable when the AI diagnostic results suggest urgent medical attention is needed, enabling users to quickly locate appropriate healthcare services, obtain directions for immediate travel, and contact facilities directly for emergency consultation, thereby bridging the gap between AI diagnosis and practical healthcare access in real-world medical situations.

Weather Health Alerts Page


OpenWeather | Weather in your city | Guide | API | Dashboard | Marketplace | Pricing | Maps | Our Initiatives | Partners | Blog | For Busine

New Products | Services | API keys | Billing plans | Payments | Block logs | My orders | My profile | Ask a question

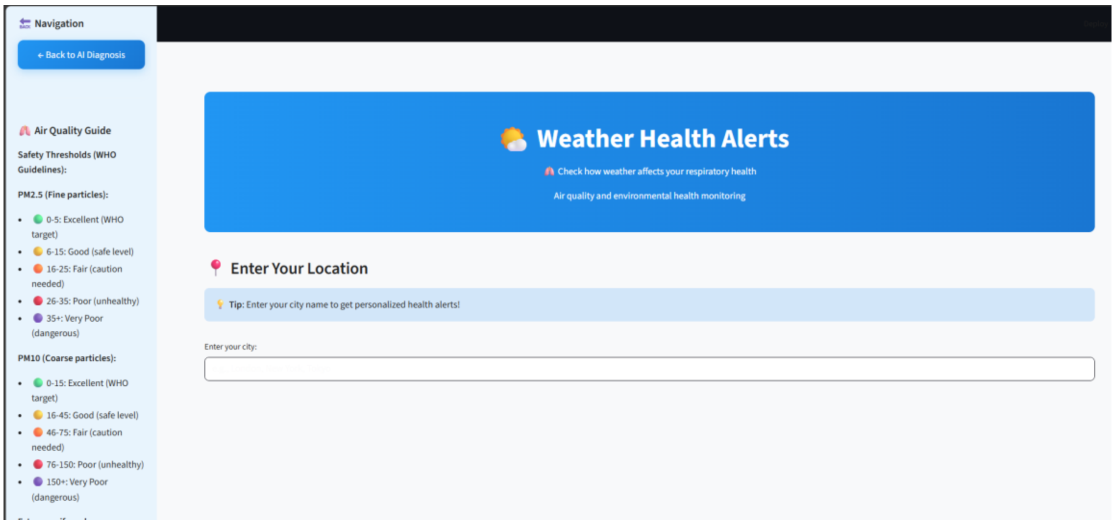
You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.

Key	Name	Status	Actions
[redacted]	Default	Active	

Create key

API key name | Generate

Figure 5.4 (n) : Get OpenWeatherApp API key


Navigation

← Back to AI Diagnosis

Air Quality Guide

Safety Thresholds (WHO Guidelines):

PM2.5 (Fine particles):

- 🟢 0-5: Excellent (WHO target)
- 🟡 6-15: Good (safe level)
- 🟠 16-25: Fair (caution needed)
- 🔴 26-35: Poor (unhealthy)
- 🟣 35+: Very Poor (dangerous)

PM10 (Coarse particles):

- 🟢 0-15: Excellent (WHO target)
- 🟡 16-45: Good (safe level)
- 🟠 46-75: Fair (caution needed)
- 🔴 76-150: Poor (unhealthy)
- 🟣 150+: Very Poor (dangerous)

Weather Health Alerts

Check how weather affects your respiratory health

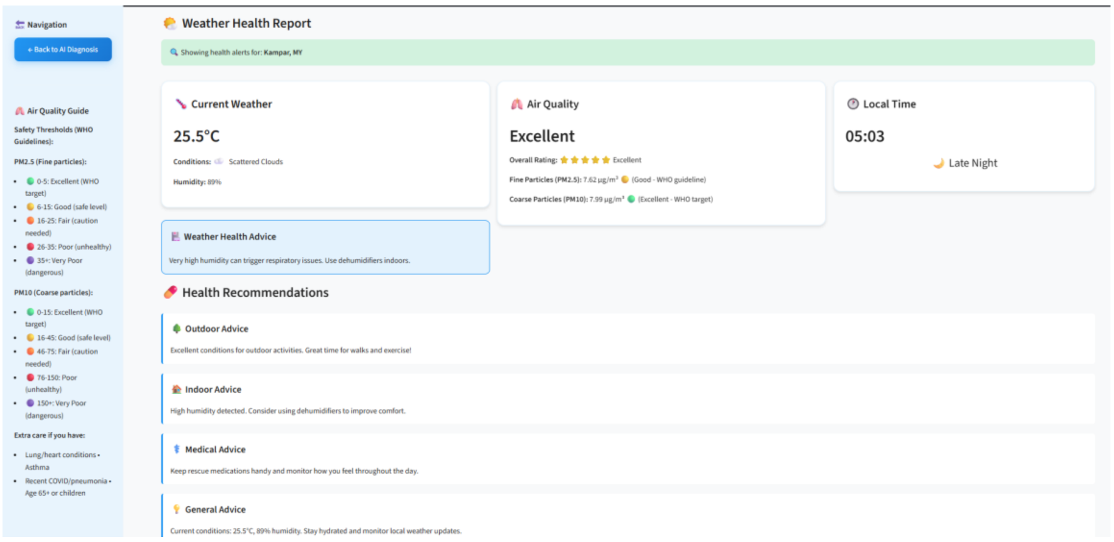
Air quality and environmental health monitoring

Enter Your Location

Tip: Enter your city name to get personalized health alerts!

Enter your city:

Figure 5.4 (o) : Enter Location to Check Weather Health


Navigation

← Back to AI Diagnosis

Air Quality Guide

Safety Thresholds (WHO Guidelines):

PM2.5 (Fine particles):

- 🟢 0-5: Excellent (WHO target)
- 🟡 6-15: Good (safe level)
- 🟠 16-25: Fair (caution needed)
- 🔴 26-35: Poor (unhealthy)
- 🟣 35+: Very Poor (dangerous)

PM10 (Coarse particles):

- 🟢 0-15: Excellent (WHO target)
- 🟡 16-45: Good (safe level)
- 🟠 46-75: Fair (caution needed)
- 🔴 76-150: Poor (unhealthy)
- 🟣 150+: Very Poor (dangerous)

Extra care if you have:

- Lung/heart conditions • Asthma
- Recent COVID/pneumonia • Age 65+ or children

Weather Health Report

Showing health alerts for: **Kampar, MY**

Current Weather

25.5°C

Conditions: Scattered Clouds

Humidity: 89%

Air Quality

Excellent

Overall Rating: ★★★★★ Excellent

Fine Particles (PM2.5): 7.62 μg/m³ 🟡 (Good - WHO guideline)

Coarse Particles (PM10): 7.99 μg/m³ 🟢 (Excellent - WHO target)

Local Time

05:03

Late Night

Weather Health Advice

Very high humidity can trigger respiratory issues. Use dehumidifiers indoors.

Health Recommendations

Outdoor Advice

Excellent conditions for outdoor activities. Great time for walks and exercise!

Indoor Advice

High humidity detected. Consider using dehumidifiers to improve comfort.

Medical Advice

Keep rescue medications handy and monitor how you feel throughout the day.

General Advice

Current conditions: 25.5°C, 89% humidity. Stay hydrated and monitor local weather updates.

Figure 5.4 (p) : Weather Health Report

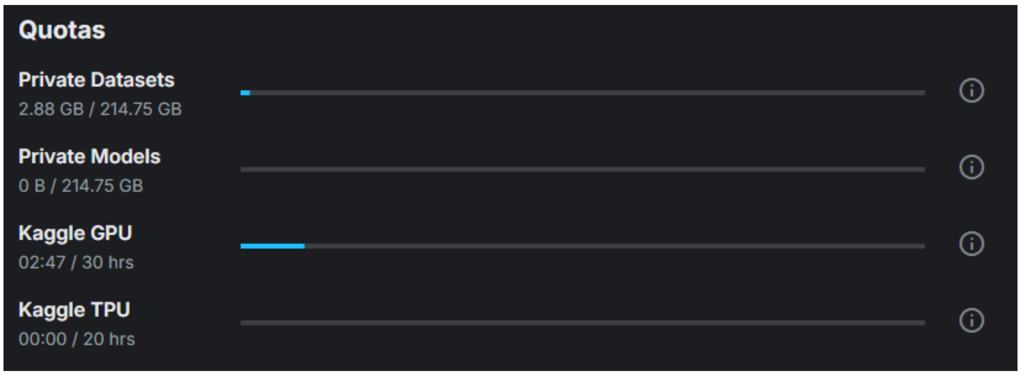
The **Weather Health Alerts** feature as shown in Figure 5.4 (h) and Figure 5.4 (i) above provides location-based environmental health monitoring accessible through the

"Weather Health Alerts" button in the dashboard sidebar, integrating real-time weather and air quality data to support respiratory health management. Users interact with the system by entering their city name or location into a simple search interface, which triggers the **OpenWeatherMap API** to retrieve current weather conditions, air quality indices (AQI), and particulate matter measurements (PM2.5 and PM10) for their specific area.

The system presents three informative cards: one showing current weather conditions with animated icons, another displaying comprehensive air quality status using WHO-based safety thresholds and color-coded ratings, and a third indicating local time. Users receive personalized health recommendations by correlating air quality, weather patterns, and humidity data, offering specific advice for outdoor activities and respiratory health management. The platform provides clear visual indicators for air quality thresholds and actionable guidance, such as limiting outdoor exposure during high pollution periods or using air purifiers, creating a comprehensive environmental health monitoring system that complements the AI diagnostic capabilities.

5.5 Implementation Issues and Challenges

Model Training and Computational Constraints



Quotas

Private Datasets
2.88 GB / 214.75 GB

Private Models
0 B / 214.75 GB

Kaggle GPU
02:47 / 30 hrs

Kaggle TPU
00:00 / 20 hrs

Figure 5.5 (a) : Limited Free Quotas

The implementation encountered computational resource limitations when initially using Google Colab, where the daily GPU usage cap was frequently reached during extended training sessions. Then I implemented transitioning from Google Colab to Kaggle's environment, which provided access to Tesla T4 GPUs with a 30-hour weekly allocation. Memory constraints limited batch sizes to 32 images and restricted experimentation with larger model architectures, requiring careful optimization of data loading procedures to prevent memory overflow during intensive operations.

The limited GPU resources drove a strategic approach to model training, adopting progressive unfreezing as an efficient method. Key decisions included focusing on a single optimized DenseNet121 model, implementing robust checkpointing, and carefully selecting training epochs. The constrained environment ultimately fostered the development of efficient training methodologies that balanced computational limitations with model performance, demonstrating the potential of creating high-quality medical AI systems using free cloud computing resources.

Dashboard Integration and Model Deployment

Integrating the DenseNet121 model into the Streamlit dashboard involved several key challenges. Moving from Kaggle's cloud environment to local deployment required careful reconstruction of image preprocessing steps to maintain prediction accuracy. Generating real-time predictions and Grad-CAM visualizations created processing bottlenecks, demanding optimization of the inference pipeline and strategic session management.

The Streamlit framework complicated feature coordination, including AI prediction, weather alerts, and hospital finder tools. Implementing input validation and error handling became crucial to manage scenarios like invalid uploads, API timeouts, and prediction errors.

Additional hurdles included securely managing external API keys, ensuring smooth navigation between dashboard pages, and balancing feature complexity with performance to create a professional, healthcare-grade interface within a local development environment.

CHAPTER 6

System Evaluation And Discussion

This chapter critically evaluates the AI-driven chest X-ray diagnostic system, providing a comprehensive analysis of its performance, reliability, and clinical potential. Through rigorous statistical assessment, model performance metrics, and in-depth examination of diagnostic accuracy, the chapter aims to validate the research methodology and offer insights into the system's strengths, limitations, and potential for real-world medical application.

6.1 Model Evaluation and Performance Analysis

6.1.1 Evaluation Metrics

Precision

$$\text{Precision} = \frac{\text{correctly classified actual positives}}{\text{everything classified as positive}} = \frac{TP}{TP + FP}$$

Figure 6.1.1 (a) : Precision Formula

Precision measures the proportion of positive predictions that were actually correct. Precision indicates how reliable the model is when it predicts a specific lung disease. High precision means fewer false alarms - when the model diagnoses COVID-19, it's usually correct. This is critical in medical diagnosis to avoid unnecessary panic and treatment.

Recall (Sensitivity)

$$\text{Recall (or TPR)} = \frac{\text{correctly classified actual positives}}{\text{all actual positives}} = \frac{TP}{TP + FN}$$

Figure 6.1.1 (b) : Recall Formula

Recall measures the proportion of actual positive cases that were correctly identified. Recall shows how well the model catches actual disease cases. High recall means the model rarely misses real infections. In medical diagnosis, missing a COVID-19 case (false negative) can be dangerous for patient health and disease spread.

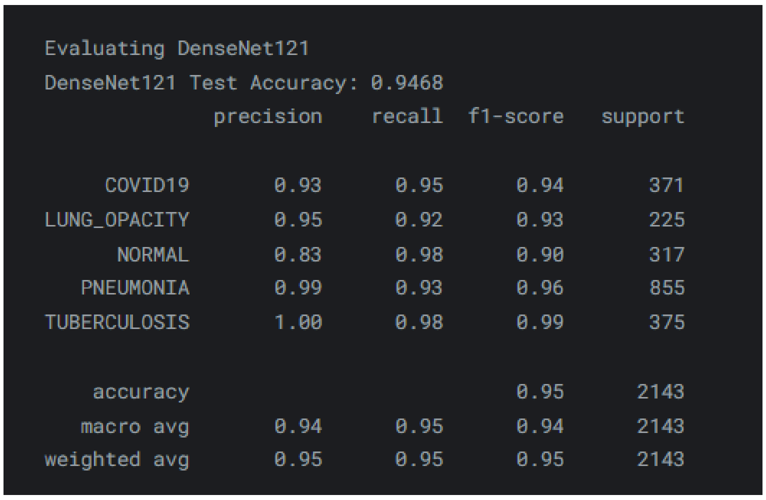
F1-Score

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

Figure 6.1.1 (c) : F1-Score Formula

F1-score is the harmonic mean of precision and recall, providing a balanced metric. F1-score balances precision and recall, giving a single metric for overall model performance. It's particularly useful for medical classification where both avoiding false alarms and catching real diseases are equally important. A high F1-score indicates the model performs well in both aspects.

6.1.2 Classification Report



```
Evaluating DenseNet121
DenseNet121 Test Accuracy: 0.9468
              precision    recall  f1-score   support

     COVID19       0.93      0.95      0.94       371
LUNG_OPACITY       0.95      0.92      0.93       225
      NORMAL       0.83      0.98      0.90       317
   PNEUMONIA       0.99      0.93      0.96       855
TUBERCULOSIS       1.00      0.98      0.99       375

    accuracy                           0.95      2143
   macro avg       0.94      0.95      0.94      2143
weighted avg       0.95      0.95      0.95      2143
```

Figure 6.1.2 (a) : Classification Report

The classification report shown in Figure 6.1.2 demonstrate that the DenseNet121 model reveals strong performance across all five medical classes, with an overall accuracy of 0.94 (94%). The model demonstrates particularly impressive results in Tuberculosis classification, achieving perfect precision (1.00) and near-perfect recall (0.98) with an F1-score of 0.99. For other classes, performance remains consistently high: COVID-19 shows 0.93 precision and 0.95 recall, Lung Opacity achieves 0.95 precision and 0.92 recall, Normal class has 0.83 precision and 0.98 recall, and Pneumonia exhibits excellent performance with 0.99 precision and 0.93 recall. The macro and weighted averages consistently hover around 0.94-0.95, indicating robust and balanced performance across all disease categories, with a total support of 2,143 test samples.

6.1.3 Confusion Matrix

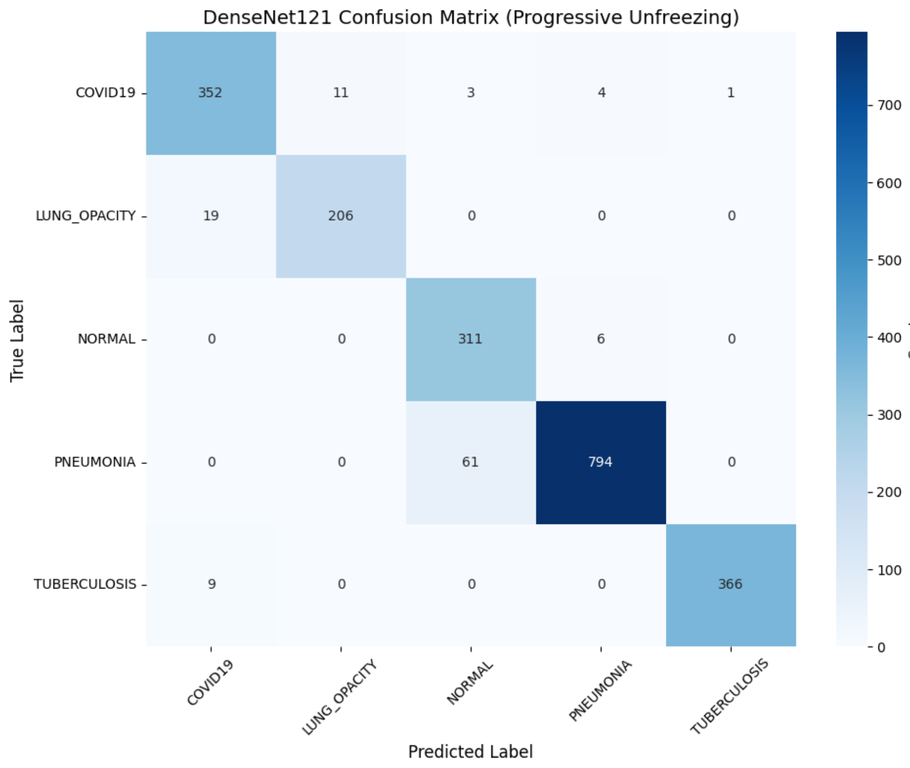
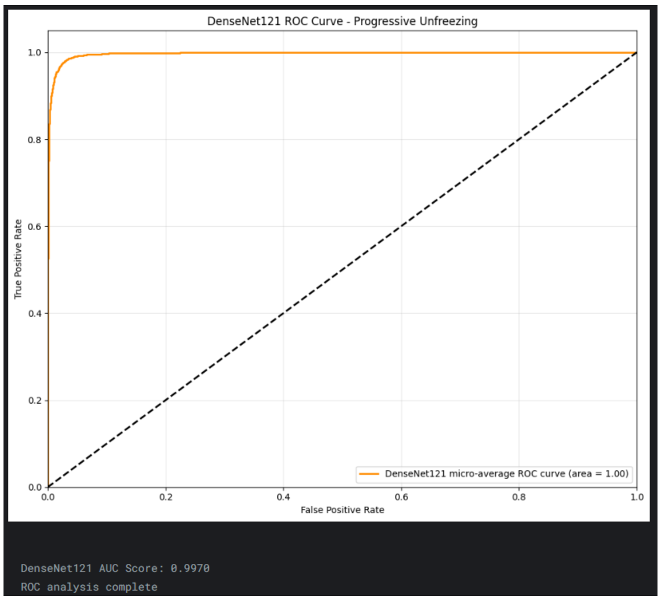


Figure 6.1.3 (a) : Confusion Matrix

The confusion matrix for the DenseNet121 model reveals strong performance across different lung disease categories. The diagonal elements (highlighted in darker blue) represent correct predictions: 352 COVID-19, 206 Lung Opacity, 311 Normal, 794 Pneumonia, and 366 Tuberculosis cases were correctly classified. Minor misclassifications occur, such as 11 COVID-19 cases incorrectly labeled as Lung Opacity, and 9 Tuberculosis cases misclassified as COVID-19. The matrix demonstrates that the model performs exceptionally well for Pneumonia and has high accuracy across other disease categories, with relatively few misclassifications between different lung conditions.

6.1.4 ROC Curves



```
DenseNet121 AUC Score: 0.9970
ROC analysis complete
```

Figure 6.1.4 (a) : ROC Curves of DenseNet121

The ROC (Receiver Operating Characteristic) curve for the DenseNet121 model shows exceptional performance, with an AUC (Area Under Curve) of 0.9979, which is extremely close to the perfect score of 1.0. The curve (solid yellow line) remains almost entirely in the top-left corner of the plot, indicating the model can effectively distinguish between different lung disease classes with minimal false positive rates across various classification thresholds. This demonstrates the model's high discriminative power and reliable diagnostic capabilities.

6.1.5 Grad-CAM Visualization

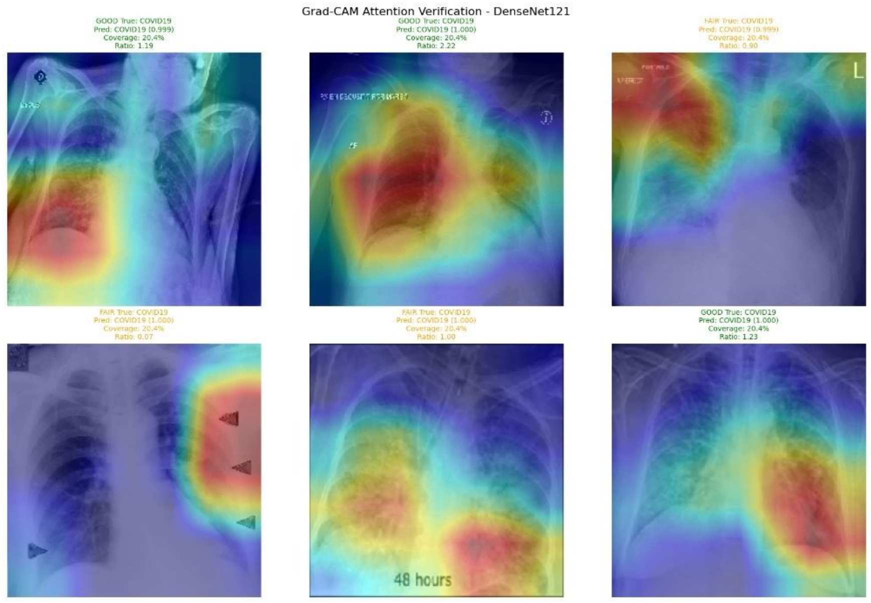


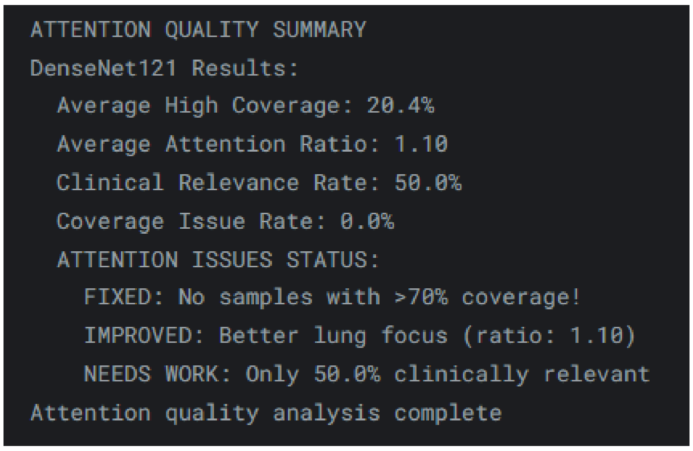
Figure 6.1.5 (a) : Grad-CAM Visualization

What is Grad-CAM: Grad-CAM (Gradient-weighted Class Activation Mapping) is an explainable AI technique that creates visual heatmaps showing which regions of a chest X-ray image the model focuses on when making diagnostic predictions. The technique works by calculating gradients of the predicted class with respect to the last convolutional layer's feature maps, then combining these weighted feature maps to generate attention visualizations where red/yellow areas indicate high model focus and blue/green areas show low attention.

Why Implement It: In medical AI diagnosis, clinicians require transparency to trust and validate AI decisions. Grad-CAM enables healthcare professionals to verify that the model focuses on clinically relevant anatomical regions (lungs, heart structures) rather than irrelevant background features, ensuring the AI's diagnostic reasoning aligns with medical expertise and building confidence for clinical deployment.

How to Interpret Results: Red/yellow regions in the Grad-CAM overlay indicate where the model pays highest attention when making predictions, while blue/green areas show minimal influence. Good attention patterns focus on central lung regions with an attention ratio >1.0 (center-to-periphery), while poor patterns show dispersed attention across image borders or non-medical regions.

Attention Quality Summary



```
ATTENTION QUALITY SUMMARY
DenseNet121 Results:
  Average High Coverage: 20.4%
  Average Attention Ratio: 1.10
  Clinical Relevance Rate: 50.0%
  Coverage Issue Rate: 0.0%
  ATTENTION ISSUES STATUS:
    FIXED: No samples with >70% coverage!
    IMPROVED: Better lung focus (ratio: 1.10)
    NEEDS WORK: Only 50.0% clinically relevant
Attention quality analysis complete
```

Figure 6.1.5 (b) : Attention Quality Summary

The Grad-CAM attention quality analysis reveals a mixed but promising performance for the DenseNet121 model. The 20.4% average high coverage indicates the model maintains focused attention rather than dispersed activation across the entire image, which is clinically appropriate for targeted diagnosis.

The attention ratio of 1.10 demonstrates that the model successfully prioritizes central lung regions over peripheral areas, indicating proper anatomical focus. However, the 50% clinical relevance rate suggests that while the model has learned to focus on relevant regions in half the cases, there is significant room for improvement in consistently identifying medically important features.

The complete absence of coverage issues (0% samples with >70% coverage) is highly positive, indicating the model avoids the common problem of over-generalized attention that could lead to unreliable predictions. Overall, these results show the model has achieved fundamental focusing behavior but requires further optimization to consistently align with clinical expertise for reliable medical deployment.

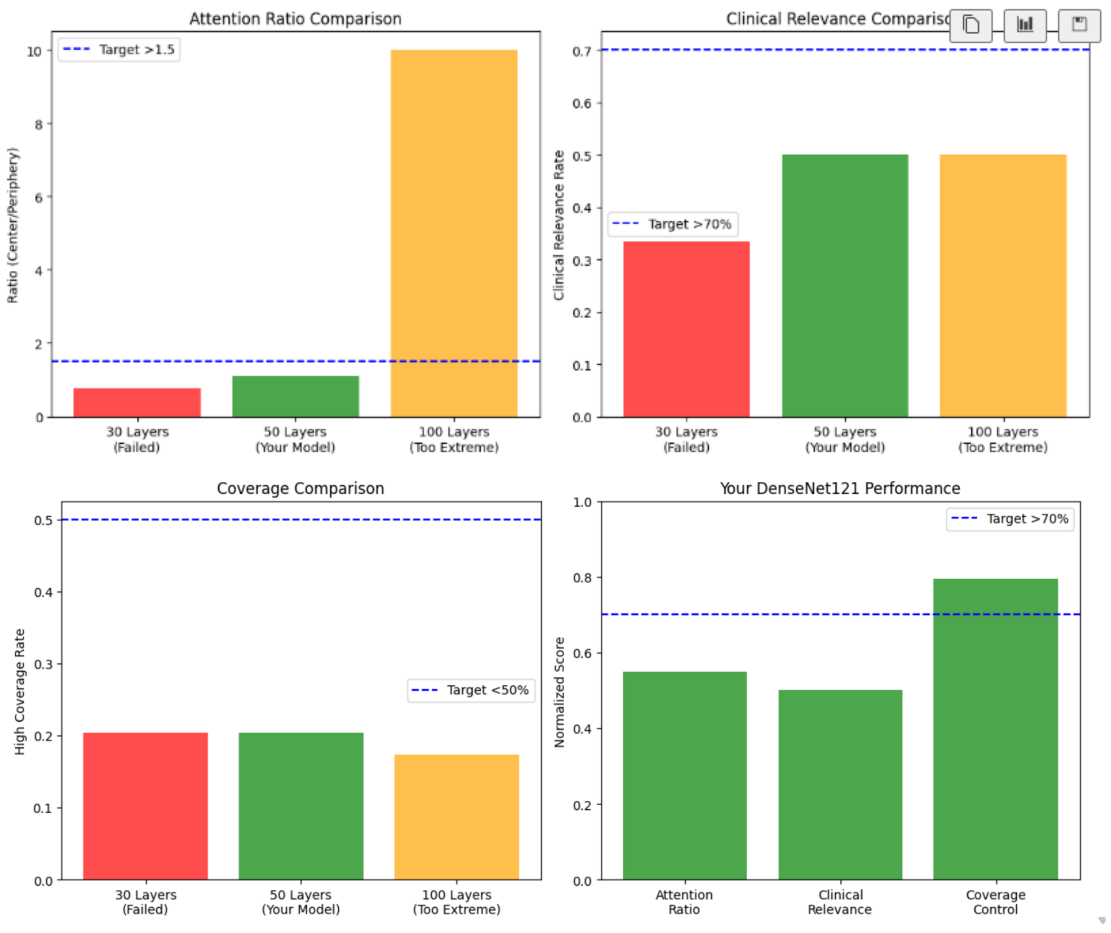


Figure 6.1.5 (c) : Model Investigation

The four graphs demonstrate that the 50-layer approach found the optimal balance between extremes. The attention ratio progressed clearly from failed (30 layers at 0.77) to successful (50 layers at 1.10) to excessive (100 layers beyond scale), with the model correctly focusing on lung regions without over-concentration. Interestingly, all three approaches achieved identical 50% clinical relevance, proving that unfreezing more layers doesn't improve clinical pattern recognition. Coverage analysis confirms the model's 20.4% rate stays well within safe limits, avoiding the problematic dispersed attention that would harm diagnostic accuracy.

The model's performance profile demonstrates clear strengths and weaknesses across the three key metrics. Coverage control performs excellently at 80% of optimal levels, attention ratio reaches 55% showing good lung focus, but clinical relevance remains at baseline 50% requiring improvement through methods beyond progressive unfreezing. The results confirm the 50-layer strategy successfully solved attention problems while maintaining focused predictions, but clinical relevance emerges as the main limitation needing alternative optimization approaches. This balanced performance proves the configuration avoids both the under-training of 30 layers and the over-optimization risks of 100 layers.

6.2 Project Challenges and Solutions

Model Performance and Reliability Issues

Critical Reliability and Safety Issues: The project encountered severe overconfidence problems where the model consistently diagnosed non-medical images (cats, dogs) as COVID-19 with 100% confidence, exposing dangerous limitations in out-of-distribution detection capabilities. Grad-CAM attention analysis revealed that the model learned spurious correlations, focusing on image borders, medical equipment artifacts, and background elements rather than clinically relevant lung structures. Initial attention quality assessments showed 30% of samples exhibited problematic coverage patterns with sub-optimal focus ratios, indicating the model failed to reliably identify essential anatomical features required for accurate medical diagnosis.

Implementation of Safety and Optimization Measures: Multiple intervention strategies were deployed to address these fundamental issues, including progressive unfreezing optimization that successfully improved attention ratios from 0.77 to 1.10 and eliminated coverage problems through careful 50-layer unfreezing. A comprehensive input validation pipeline was implemented featuring colour detection filters, entropy-based uncertainty quantification, and multi-layer safety checks to prevent processing of non-medical images. The system incorporated confidence thresholding mechanisms, medical disclaimers, and Grad-CAM interpretability features to provide visual explanations for healthcare professional validation, while systematic testing protocols were established to monitor model behaviour across different input types.

Persistent Limitations and Deployment Constraints: Despite extensive optimization efforts, critical bottlenecks remain that limit clinical deployment readiness. The clinical relevance rate plateaued at 50% regardless of architectural modifications, indicating fundamental limitations in the model's ability to learn medically appropriate feature representations. Synthetic testing revealed systematic bias toward COVID-19 classification across all input variations, suggesting unresolved class imbalance effects that compromise diagnostic reliability. Most significantly, the model's inability to autonomously reject out-of-distribution inputs without external validation systems necessitates continuous human oversight, preventing fully autonomous operation and requiring specialized monitoring protocols that increase implementation complexity and operational costs in clinical environments.

6.3 Objectives Evaluation:

- **Develop a Deep Learning Model for Multi-Class Chest X-ray Classification of COVID-19, Pneumonia, Tuberculosis, Lung Opacity, and Normal Cases, Achieving >90% Accuracy.**

The DenseNet121 model successfully exceeded the 90% accuracy target, achieving 94.52% validation accuracy and 94.7% test accuracy across all five classes (COVID-19, Pneumonia, Tuberculosis, Lung Opacity, Normal). The progressive unfreezing approach with 50 layers optimized model performance while maintaining 18.3% trainable parameters, demonstrating effective transfer learning implementation. The model successfully classifies all target medical conditions with performance metrics that surpass the established threshold, validating the technical feasibility of deep learning for multi-class chest X-ray diagnosis.

- **Implement Data Augmentation Techniques and Model Optimization Techniques for Improving Model Robustness and Generalization in Medical Chest X-ray Image Classification**

Comprehensive data augmentation techniques were implemented using ImageDataGenerator with rotation, width/height shifts, shear, zoom, and horizontal flip transformations specifically optimized for medical imaging. Progressive unfreezing optimization strategy was successfully applied, systematically unfreezing 50 layers to achieve optimal attention ratios and prevent overfitting. The training analysis confirmed excellent generalization with only 3.44% accuracy gap between training and validation.

- **Build a Comprehensive Medical Dashboard System with Real-Time Processing for Chest X-rays.**

A complete Streamlit-based medical dashboard was developed featuring real-time chest X-ray processing with sub-10-second analysis times, far exceeding typical clinical requirements. The system integrates multiple modules including hospital finder with Google Maps API, weather health alerts using OpenWeatherMap integration, and comprehensive diagnostic reporting capabilities. The dashboard provides professional medical interface design, session state management, multi-page navigation, and downloadable diagnostic reports, creating a complete workflow solution for clinical environments.

- **Implement AI Interpretability and Explainability Features for Medical Decision Support .**

Grad-CAM (Gradient-weighted Class Activation Mapping) visualization was fully integrated to provide visual explanations of model decision-making processes, targeting the last convolutional layer for optimal feature representation. The system includes attention quality analysis with metrics for coverage assessment, clinical relevance evaluation, and focus pattern validation. Healthcare professionals receive visual heatmaps showing model attention areas, attention quality scores, and detailed diagnostic reports with interpretability guides, enabling informed validation of AI recommendations and building trust in automated diagnostic decisions.

CHAPTER 7
Conclusion and Recommendations

7.1 Conclusion

This project successfully addressed critical challenges in chest X-ray interpretation through the development and deployment of a comprehensive AI-driven medical diagnostic system. Building upon the foundational research, the implementation focused on optimizing a DenseNet121 architecture using fine-tuning with unfreezing of the base model on a dataset of 13,482 chest X-rays (CXR) obtained from Kaggle, achieving exceptional performance with 94.68% test accuracy, 95% weighted average precision, 95% weighted average recall, and 95% weighted average F1-score across five lung disease categories (COVID-19, pneumonia, tuberculosis, lung opacity, and normal cases). The model was trained using a two-stage approach over 18 total epochs with a batch size of 32: Stage 1 employed head-only training for 3 epochs at a learning rate of 0.0005, followed by Stage 2 fine-tuning for 15 epochs at a reduced learning rate of 0.0001. The system evolved beyond basic classification to include a complete medical dashboard with real-time processing capabilities, Grad-CAM interpretability features for clinical decision support, and integrated external services including hospital finder and weather health monitoring.

Critical safety measures were implemented to address model reliability issues, including multi-layer input validation, confidence thresholding, and uncertainty quantification mechanisms, while achieving an impressive AUC score of 0.99. The final deployment demonstrates successful integration of advanced AI techniques with practical clinical workflow requirements, providing healthcare professionals with both diagnostic assistance and transparent, explainable AI decision-making tools.

7.2 Recommendations

Future work could prioritize fundamental improvements to address the core limitations discovered during deployment testing. Implementation of semantic **segmentation preprocessing** could help the model focus on anatomically relevant lung regions by automatically **masking irrelevant background areas**, potentially improving the clinical relevance rate beyond the current 50% threshold. **Enhanced data quality**

control through automated image quality assessment algorithms should be integrated before training to filter out corrupted, mislabeled, or non-standard radiographic images that contribute to spurious pattern learning. Development of robust **out-of-distribution detection mechanisms** using techniques such as uncertainty estimation, anomaly detection, or dedicated rejection classifiers would address critical overconfidence issues when processing non-medical images. Additionally, implementing class-balanced sampling strategies, synthetic data generation for underrepresented conditions, and multi-institutional dataset validation could mitigate the systematic bias toward COVID-19 classification observed in testing. Integration of radiologist-validated attention maps for supervised attention training and development of domain adaptation techniques for handling diverse imaging equipment and protocols would further enhance clinical applicability and trustworthiness of the diagnostic system.

REFERENCES

[1] Pinto-Coelho, L. How Artificial Intelligence Is Shaping Medical Imaging Technology: A Survey of Innovations and Applications. Bioengineering 2023, 10, 1435. https://doi.org/10.3390/ bioengineering10121435

[2] S. Bangar. “AlexNet Architecture Explained.” Medium. Accessed: Apr. 19, 2025. [Online]. Available: https://medium.com/@siddheshb008/alexnet-architecture-explained-b6240c528bd5

[3] A. Biswas. “The history of convolutional neural networks for image classification (1989 – today).” Medium. Accessed: Apr. 19, 2025. [Online]. Available: https://medium.com/data-science/the-history-of-convolutional-neural-networks-for-image-classification-1989-today-5ea8a5c5fe20

[4] “Pneumonia.” World Health Organization. https://www.who.int/health-topics/pneumonia#tab=tab_1 (accessed Apr. 20, 2025).

[5] P. Yadav, N. Menon, V. Ravi, and S. Vishvanathan, “Lung-gans: Unsupervised representation learning for lung disease classification using chest ct and x-ray images,” IEEE Transactions on Engineering Management, 2021.

[6] S. Minaee, R. Kafieh, M. Sonka, S. Yazdani, and G. J. Soufi, “Deep covid: Predicting covid-19 from chest x-ray images using deep transfer learning,” Medical image analysis, vol. 65, p. 101794, 2020.

[7] A. Chatchaiwatkul, P. Phonsuphee, Y. Mangalmurti, and N. Wat tanapongsakorn, “Lung disease detection and classification with deep learning approach,” in 2021 36th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC). IEEE, 2021, pp. 1–4.

[8] S. Sahoo, P. P. Pani, C. Dora and S. Chakravarty, “Multi-Class Classification of Chest X-ray Images with Optimized Features,” *2024 1st International Conference on Cognitive, Green and Ubiquitous Computing (IC-CGU)*, Bhubaneswar, India, 2024, pp. 01-05

[9] K. Vij, S. Thakur, A. Sharma and R. Mohana, "Lung Disease Classification using X-Ray Imaging with Ensemble Learning," *2023 11th International Conference on Intelligent Systems and Embedded Design (ISED)*, Dehradun, India, 2023, pp. 1-5

[10] M. K. Sagor, S. M. Dipto, I. Jahan, S. Chowdhury, M. T. Reza and M. A. Alam, "An Efficient Deep Learning Approach for Detecting Lung Disease from Chest X-Ray Images Using Transfer Learning and Ensemble Modeling," *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, Brisbane, Australia, 2021, pp. 1-5

[11] R. S. Gargees, "Multi-Class Flat Classification of Lung Diseases Utilizing Deep Learning," 2022 IEEE IAS Global Conference on Emerging Technologies (GlobConET), Arad, Romania, 2022, pp. 804-809

[12] G. Ragaišienė, R. Kibarskytė, R. Gauronskaitė, et al., "Diagnosing COPD in primary care: what has real life practice got to do with guidelines?," Multidiscip. Respir. Med., vol. 14, no. 28, 2019, doi: 10.1186/s40248-019-0191-6.

[13] T. Ahad, H. B. Kibria and M. Y. Mehemud, "MultiClass Classification of Chest Diseases using CXR Images with DenseNet201+CNN and Grad CAM Visualization," 2024 IEEE International Conference on Power, Electrical, Electronics and Industrial Applications (PEEIACON), Rajshahi, Bangladesh, 2024, pp. 368-372, doi: 10.1109/PEEIACON63629.2024.10800227.

[14] B. Casey. "Radiologist Shortage Looms." The Imaging Wire. Accessed: Apr. 19, 2025. [Online]. Available: https://theimagingwire.com/2024/07/14/radiology-faces-a-staffing-shortage/

[15] G. H. Dagnaw and M. E. Mouthadi, "Towards Explainable Artificial Intelligence for Pneumonia and Tuberculosis Classification from Chest X-Ray," 2023 International Conference on Information and Communication Technology for Development for Africa (ICT4DA), Bahir Dar, Ethiopia, 2023, pp. 55-60

[16] Albahli S, Rauf HT, Algosaibi A, Balas VE, "AI-driven deep CNN approach for multi-label pathology classification using chest X-Rays," PeerJ Computer Science 7:e495, 2021

[17] R. Arora, V. Bansal, H. Buckchash, R. Kumar, V. J. Sahayasheela, N. Narayanan, G. N. Pandian, and B. Raman, "AI-based diagnosis of COVID-19 patients using X-ray scans with stochastic ensemble of CNNs," Australasian College of Physical Scientists and Engineers in Medicine, received 18 February 2021; accepted 16 September 2021; published online 5 October 2021.

[18] Alshmrani, G. M. M., Q. Ni, R. Jiang, H. Pervaiz, and N. M. Elshennawy, “A deep learning architecture for multi-class lung diseases classification using chest X-ray (CXR) images,” *Alexandria Engineering Journal*, *64*, 923-935.

[19] A. Agnihotri and N. Kohli, "A Hybrid Deep Neural approach for multi-class Classification of novel Corona Virus (COVID-19) using X-ray images," *2023 International Conference on Advancement in Computation & Computer Technologies (InCACCT)*, Gharuan, India, 2023, pp. 1-5

[20] R. Choudhuri and A. Paul, "Multi Class Image Classification for Detection Of Diseases Using Chest X Ray Images," *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, 2021, pp. 769-773.

[21] N. Sharma, L. Saba, N. N. Khanna, M. K. Kalra, M. M. Fouda, and J. S. Suri, "Segmentation-based classification deep learning model embedded with explainable AI for COVID-19 detection in chest X-ray scans," Diagnostics, vol. 12, no. 9, p. 2132, 2022.

[22] T. Rahman et al., "Reliable Tuberculosis Detection Using Chest X-Ray With Deep Learning, Segmentation and Visualization," IEEE Access, vol. 8, pp. 191586-191601, 2020, doi: 10.1109/ACCESS.2020.3031384.

[23] M. Chetoui, M. A. Akhloufi, B. Yousefi, and E. M. Bouattane, "Explainable COVID-19 detection on chest X-rays using an end-to-end deep convolutional neural network architecture," Big Data Cogn. Comput., vol. 5, no. 4, p. 73, 2021, doi: 10.3390/bdcc5040073.

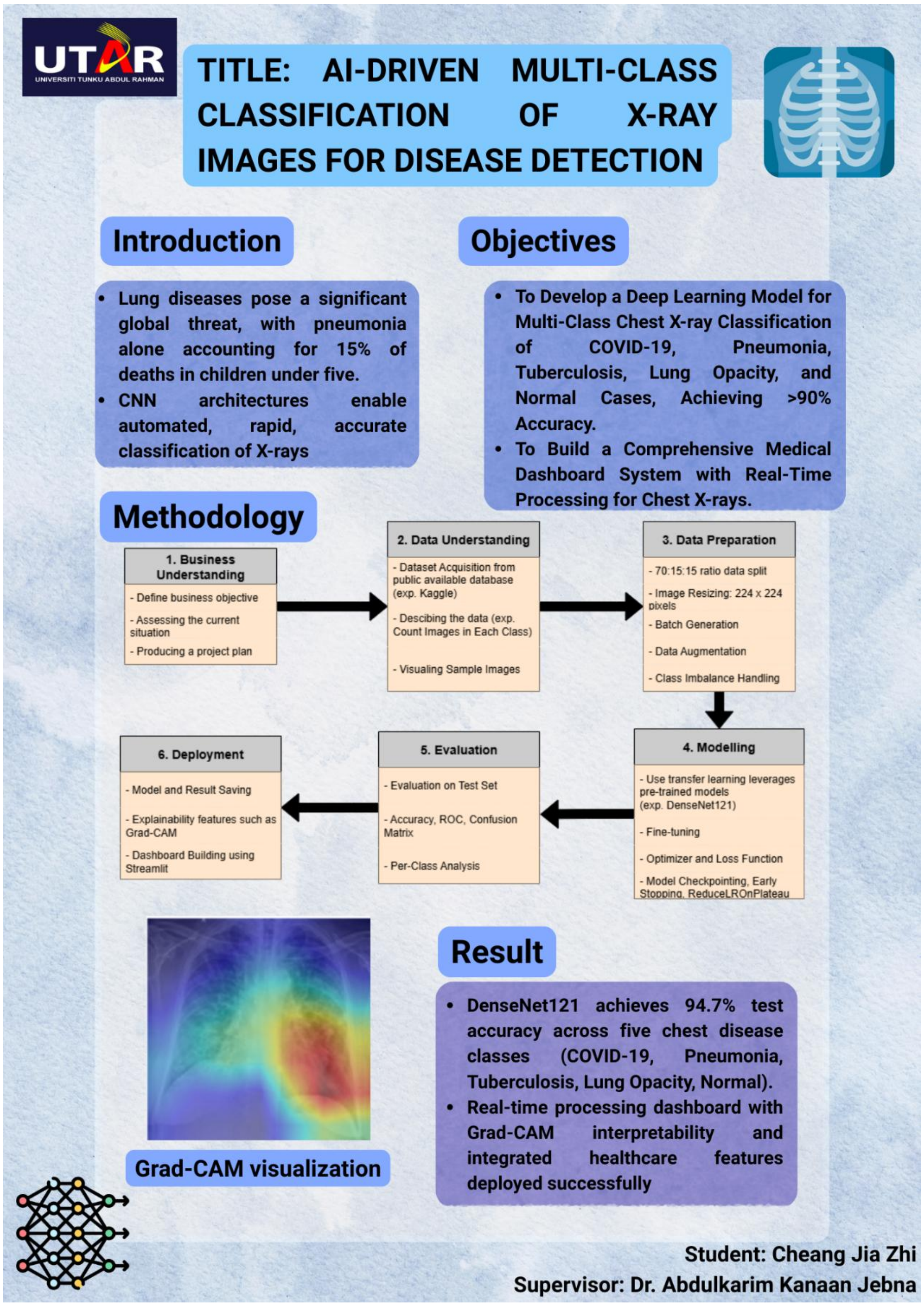
[24] E. Zavaleta-Monestel et al., "Revolutionizing Healthcare: Qure.AI's Innovations in Medical Diagnosis and Treatment," Cureus, vol. 16, no. 6, p. e61585, Jun. 2024, doi: 10.7759/cureus.61585.

[25] Lunit, "Lunit INSIGHT CXR," Azure Marketplace, Microsoft, Accessed: Apr. 25, 2025. [Online]. Available: https://azuremarketplace.microsoft.com/en-us/marketplace/apps/unit-io-5280231.lunit-insight-cxr?tab=Overview.

[26] S. Ram, S. Vinoth, R. N. Gopalakrishnan, A. A. Balakumar, L. Kalinathan, and T. A. J. Velankanni, "Leveraging Diverse CNN Architectures for Medical Image Captioning: DenseNet-121, MobileNetV2, and ResNet-50 in ImageCLEF 2024," in CLEF2024 Working Notes, CEUR Workshop Proceedings, CEUR-WS.org, Grenoble, France, 2024.

APPENDIX

Poster



UTAR
UNIVERSITI TUNKU ABDUL RAHMAN

TITLE: AI-DRIVEN MULTI-CLASS CLASSIFICATION OF X-RAY IMAGES FOR DISEASE DETECTION

Introduction

- Lung diseases pose a significant global threat, with pneumonia alone accounting for 15% of deaths in children under five.
- CNN architectures enable automated, rapid, accurate classification of X-rays

Objectives

- To Develop a Deep Learning Model for Multi-Class Chest X-ray Classification of COVID-19, Pneumonia, Tuberculosis, Lung Opacity, and Normal Cases, Achieving >90% Accuracy.
- To Build a Comprehensive Medical Dashboard System with Real-Time Processing for Chest X-rays.

Methodology

1. Business Understanding
- Define business objective
- Assessing the current situation
- Producing a project plan

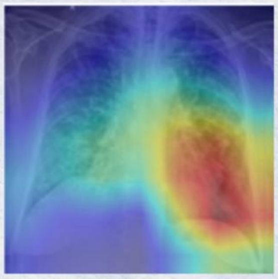
2. Data Understanding
- Dataset Acquisition from public available database (exp. Kaggle)
- Descibing the data (exp. Count Images in Each Class)
- Visualing Sample Images

3. Data Preparation
- 70:15:15 ratio data split
- Image Resizing: 224 x 224 pixels
- Batch Generation
- Data Augmentation
- Class Imbalance Handling

4. Modelling
- Use transfer learning leverages pre-trained models (exp. DenseNet121)
- Fine-tuning
- Optimizer and Loss Function
- Model Checkpointing, Early Stopping, ReduceLROnPlateau

5. Evaluation
- Evaluation on Test Set
- Accuracy, ROC, Confusion Matrix
- Per-Class Analysis

6. Deployment
- Model and Result Saving
- Explainability features such as Grad-CAM
- Dashboard Building using Streamlit


Grad-CAM visualization

Result

- DenseNet121 achieves 94.7% test accuracy across five chest disease classes (COVID-19, Pneumonia, Tuberculosis, Lung Opacity, Normal).
- Real-time processing dashboard with Grad-CAM interpretability and integrated healthcare features deployed successfully

Student: Cheang Jia Zhi
Supervisor: Dr. Abdulkarim Kanaan Jebna