An
Industry Oriented Mini Project Report
on

MEDICAL IMAGE CLASSIFICATION USING DEEP LEARNING

Submitted in partial fulfillment of the requirements for the award of degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

S. AKSHAYA 227Z1A05F9

Under the Guidance of
Dr.A.Prashanthi
Associate Professor

SCHOOL OF ENGINEERING
Department of Computer Science and Engineering

NALLA NARASIMHA REDDY
EDUCATION SOCIETY'S GROUP OF INSTITUTIONS
(AN AUTONOMOUS INSTITUTION)

Approved by AICTE, New Delhi, Chowdariguda (V) Korremula 'x'Roads, via Narapally, Ghatkesar (Mandal) Medchal (Dist), Telangana-500088
2024-2025

NALLA NARASIMHA REDDY
Education Society's Group of Institutions - Integrated Campus
(UGC AUTONOMOUS INSTITUTION)

SCHOOL OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project report titled **"MEDICAL IMAGE CLASSIFICATION USING DEEP LEARNING"** is being submitted by **S. Akshaya (227Z1A05F9)** in Partial fulfillment for the award of **Bachelor of technology in Computer Science & Engineering** is a record bonafide work carried out by her. The results embodied in this report have not been submitted to any other University for the award of any degree.

Internal Guide
(Dr.A.Prashanthi)

Head of the Department
(Dr.K.Rameshwaraiah)

Submitted for the University Examination held on…….

External Examiner

DECLARATION

I S. Akshaya, the student of **Bachelor of Technology in Computer Science and Engineering, Nalla Narasimha Reddy Education Society's Group of Institutions**, Hyderabad, Telangana, hereby declare that the work presented in this project work entitled **MEDICAL IMAGE CLASSIFICATION USING DEEP LEARNING** is the outcome of my own bonafide work and is correct to the best of my knowledge and this work has been undertaken taking care of engineering ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning.

S. Akshaya 227Z1A05F9

Date:

Signature:

ACKNOWLEDGEMENT

I express my sincere gratitude to our guide **Dr. A. Prashanthi,** Associate Professor, Department of Computer Science and Engineering, NNRESGI, who motivated throughout the period of the project and also for her valuable and intellectual suggestions, guidance, and constant encouragement right throughout my work.

I would like to express my profound gratitude to my project coordinator **Mr. T. Madhu**, Associate Professor, Department of Computer Science and Engineering, NNRESGI, for his support and guidance in completing my project and for giving me this opportunity to present the project work.

I profoundly express thanks to **Dr. K. Rameshwaraiah,** Professor & Head, Department of Computer Science and Engineering, NNRESGI, for his cooperation and encouragement in completing the project successfully.

I wish to express my sincere thanks to **Dr. G. Janardhana Raju,** Dean School of Engineering, NNRESGI, for providing the facilities for completion of the project.

I wish to express our sincere thanks to **Dr**. **C. V. Krishna Reddy,** Director, NNRESGI, for providing the facilities for completion of the project.

Finally, I would like to thank Project Review Committee (PRC) members, all the faculty members and supporting staff, Department of Computer Science and Engineering, NNRESGI, for extending their help in all circumstances.

By

S. Akshaya 227Z1A05F9

ABSTRACT

Medical image classification is an important task in healthcare that helps doctors analyze medical images like X-rays, MRI scans, and CT scans. It involves using computer programs and deep learning models, especially Convolutional Neural Networks (CNNs), to automatically detect and classify diseases in medical images. This technology helps in early diagnosis, treatment planning, and monitoring diseases, reducing the need for manual interpretation by doctors. By using large medical datasets and advanced machine learning techniques, medical image classification improves accuracy and decision-making in healthcare. Researchers are now working on making these models more interpretable, generalizable, and easier to use in real hospitals, ensuring they can be effectively integrated into real-world clinical practice**.**

Keywords: Medical Image Classification, Deep Learning, CNNs, Medical Imaging, Automated Diagnosis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

S. No.	Abbreviation	Definitions
1	CNN	Convolutional Neural Network
2	Grad-CAM	Gradient-weighted Class Activation Mapping
3	HDF5 (.h5)	Hierarchical Data Format version 5
4	API	Application Programming Interface
5	ReLU	Rectified Linear Unit

1. INTRODUCTION

Medical image classification is a vital area of research in healthcare that involves using advanced computational techniques to analyze and categorize medical images such as X-rays, CT scans, and MRIs. With the increasing demand for accurate and timely diagnosis, especially in critical conditions, automated image classification helps support clinicians in making informed decisions. Deep learning, particularly convolutional neural networks (CNNs), has revolutionized this domain by enabling models to learn and detect patterns in medical images with high precision. One practical example is the classification of pneumonia from chest X-ray images, where early detection can significantly improve patient treatment outcomes. By integrating AI into diagnostic workflows, medical image classification aims to enhance accuracy, reduce workload on radiologists, and expand access to quality healthcare.

1.1 Motivation

Medical image classification is a rapidly advancing field aimed at enhancing diagnostic accuracy and efficiency in healthcare. With the growing number of patients and limited access to specialized radiologists, especially in remote areas, there is a critical need for automated tools that can assist in interpreting complex medical images. Deep learning, particularly convolutional neural networks (CNNs), has emerged as a powerful solution by enabling models to learn visual patterns directly from images. One such application is pneumonia detection using chest X-rays, where timely diagnosis is crucial to prevent severe health complications. Automated classification systems can provide consistent, accurate results, reduce diagnostic time, and support clinical decision-making. The motivation behind this project is to harness the capabilities of AI to bridge the healthcare gap, improve patient outcomes, and demonstrate how interpretable tools like Grad-CAM can enhance trust in AI-assisted diagnoses.

1.2 Problem Definition

The primary problem in medical image classification is the accurate and reliable identification of diseases from medical imaging data such as X-rays, MRIs, and CT scans. Manual analysis by radiologists is time-consuming, subject to human error, and often limited by resource constraints, especially in remote or under-resourced areas. Developing an automated system using deep learning models like convolutional neural networks (CNNs) aims to classify medical images efficiently and accurately. For instance, in pneumonia detection, the challenge is to differentiate between normal and infected lungs using chest X-rays with high sensitivity and specificity. The goal is to build a model that can assist in early diagnosis, reduce misdiagnosis, and support healthcare professionals in making quick and informed clinical decisions.

1.3 Objective of Project

The objective of a project focused on medical image classification using deep learning is to develop advanced algorithms that can automatically analyze and categorize medical images, such as X-rays, MRIs, and CT scans, thereby enhancing diagnostic accuracy and efficiency. By leveraging convolutional neural networks (CNNs) and large annotated datasets, the project aims to facilitate early disease detection, reduce the workload on radiologists, and improve patient outcomes through more accurate diagnoses. Additionally, the goal is to create user-friendly interfaces that can be seamlessly integrated into clinical workflows, ultimately contributing to the ongoing research in medical imaging and artificial intelligence, and paving the way for future innovations in healthcare technology.

1.4 Limitations of Project

One major limitation of medical image classification using deep learning is the reliance on large, high-quality datasets. In the medical field, such datasets are often difficult to obtain due to privacy regulations, limited expert annotations, and potential biases in data collection. As a result, models may not generalize well across diverse populations or clinical environments.

Another challenge is the high computational cost associated with training and deploying deep learning models. These systems often require powerful GPUs or cloud-based infrastructure, which may not be accessible in many hospitals or rural healthcare settings. This restricts the practical use of these technologies in real-time clinical workflows.

Finally, variability in imaging techniques such as differences in equipment, resolution, or protocols can affect model accuracy. Ethical concerns also arise, including patient data privacy, the risk of algorithmic bias, and accountability in decision-making. Integrating AI tools into existing healthcare systems requires careful planning, validation, and training to ensure safe and effective use.

1.4.2 Data Availability and Imbalance

The model's performance heavily depends on the availability of large, well-labeled, and balanced datasets. In many cases, medical datasets are limited or skewed toward one class (e.g., more pneumonia cases than normal).

1.4.2 Generalization

Trained model might not perform well on unseen data from different hospitals or imaging equipment due to variations in image quality, resolution, or noise.

1.4.3 Model Interpretability

Although tools like Grad-CAM provide visual insights, deep learning models remain largely black-box systems, making it difficult to interpret the decision-making process in critical medical applications.

1.4.4 False Predictions

Even with high accuracy, there is a risk of false positives (predicting disease when there is none) or false negatives (missing the disease), which can have serious consequences in a medical context.

1.4.5 Limited Scope

This project focuses specifically on pneumonia detection from chest X-rays and may not generalize to other diseases or imaging modalities without significant retraining and validation.

1.4.6 Lack of Clinical Validation

The system has not been validated in real clinical environments, and its use should be limited to educational or experimental settings until approved by medical authorities.

2 LITERATURE SURVEY

2.1 Introduction

Researchers have focused on harnessing the power of deep learning to revolutionize the field of medical image classification, driven by the growing availability of medical imaging data and the need for faster, more accurate diagnostic tools. Traditional machine learning methods required manual feature engineering, which is both time-consuming and heavily reliant on domain expertise. However, with the advent of Convolutional Neural Networks (CNNs), researchers discovered the potential to automatically learn and extract complex hierarchical features from raw image data, making CNNs exceptionally well-suited for medical imaging tasks. A significant body of work has explored the application of CNNs to detect and classify various conditions including pneumonia, lung cancer, diabetic retinopathy, and brain tumors, with chest X-rays emerging as a widely studied modality due to their affordability and diagnostic relevance. In particular, pneumonia detection has been a popular use case, with studies utilizing models like AlexNet, VGGNet, ResNet, and MobileNet, often fine-tuned through transfer learning to perform with high accuracy on relatively small medical datasets. Researchers have also investigated hybrid approaches combining CNNs with attention mechanisms and explainable AI techniques such as Grad-CAM to make the diagnostic process more transparent and interpretable for clinicians. Despite these advancements, challenges persist in the form of data scarcity, annotation quality, class imbalance, model generalization, and the need for regulatory approval in clinical environments. Nevertheless, the literature reflects a clear trajectory toward building robust, interpretable, and clinically integrated AI systems that can augment human expertise and ultimately improve patient outcomes. This project builds upon these foundations by implementing a MobileNetV2-based classifier for pneumonia detection, integrated with Grad-CAM visualization, thereby contributing to the development of accessible and explainable medical AI solutions.

1.5Existing System

Researchers and developers have made significant strides in applying machine learning and deep learning techniques to medical image classification. Traditionally, systems relied on manual feature extraction, where domain experts designed handcrafted features based on textures, intensities, and shapes found in medical images such as X-rays, CT scans, or MRIs. These features were then fed into classical classifiers like Support Vector Machines (SVMs), Decision Trees, or Random Forests. While these systems offered some success, their performance heavily depended on the quality and relevance of the manually selected features, which often failed to generalize across different datasets or imaging modalities.

The introduction of deep learning, particularly Convolutional Neural Networks (CNNs), marked a major evolution in the field. CNNs automatically extract features from image data through multiple hierarchical layers, significantly reducing the dependency on human-crafted inputs. Pre-trained models like VGG16, ResNet50, InceptionV3, and MobileNetV2, often fine-tuned through transfer learning, have been widely adopted for tasks such as pneumonia detection from chest X-ray images. These models have shown remarkable accuracy and robustness compared to traditional methods, especially when trained on large datasets like ChestX-ray14 or the NIH dataset.

However, existing systems are not without limitations. Many models lack interpretability, making it difficult for clinicians to trust and adopt them in diagnostic workflows. As a result, explainability tools like Grad-CAM and saliency maps have been incorporated to visualize model attention and build trust. Another concern lies in the quality of training data—many models are trained on imbalanced or limited datasets, leading to potential biases and reduced generalization in real-world settings. Moreover, integration with clinical systems remains a challenge, requiring considerations for latency, regulatory approval, and data privacy.

Despite these challenges, existing systems lay a solid foundation for more robust, interpretable, and clinically viable AI-based diagnostic tools. Continuous research is focused on enhancing model performance, explainability, and adaptability to diverse clinical environments.

2.2.1 Constraint of Existing System

- **Lack of Interpretability**: Many deep learning models, particularly CNNs, operate as "black boxes," making it difficult for clinicians to understand how decisions are made. This lack of transparency can hinder trust and acceptance among healthcare professionals.
- **Data Quality and Imbalance**: Many models are trained on datasets that are either limited in size or imbalanced, leading to potential biases. For instance, if a dataset contains significantly more images of one condition than another, the model may perform poorly on underrepresented conditions.
- **Generalization Issues**: Models trained on specific datasets may not generalize well to other datasets or imaging modalities. Variability in imaging techniques, patient demographics, and disease presentations can affect model performance in real-world applications.
- **Integration Challenges**: Integrating AI systems into existing clinical workflows can be complex. Issues such as latency in processing, compatibility with existing software, and the need for regulatory approval can impede the seamless adoption of these technologies.
- **Data Privacy and Security**: Handling sensitive medical data raises concerns about patient privacy and data security. Compliance with regulations such as HIPAA (Health Insurance Portability and Accountability Act) is essential but can complicate data sharing and model training.
- **Resource Intensity**: Training deep learning models often requires significant computational resources and expertise. This can be a barrier for smaller healthcare institutions that may lack the necessary infrastructure or personnel.
- **Dependence on Large Datasets**: While deep learning models perform well with large datasets, acquiring sufficient high-quality labeled data can be challenging. This is particularly true in specialized medical fields where data may be scarce.

- **Overfitting**: Deep learning models are prone to overfitting, especially when trained on small or non-representative datasets. This can lead to models that perform well on training data but poorly on unseen data.
- **Evolving Medical Knowledge**: The field of medicine is constantly evolving, and models trained on historical data may not account for the latest research or treatment protocols. Continuous updates and retraining are necessary to maintain relevance.
- **User Acceptance**: Clinicians may be resistant to adopting AI tools due to concerns about job displacement, reliance on technology, or skepticism about the accuracy of AI-generated diagnoses.

2.3 Proposed System

The proposed system aims to enhance the efficiency, accessibility, and interpretability of pneumonia detection using a deep learning-based diagnostic framework. With the rise in respiratory illnesses and the need for rapid screening tools, especially in resource-constrained healthcare environments, this system leverages advanced artificial intelligence techniques to assist medical professionals in interpreting chest X-ray images. The system integrates a lightweight convolutional neural network architecture, MobileNetV2, and an interpretability method, Grad-CAM (Gradient-weighted Class Activation Mapping), into a user-friendly web-based interface built with Streamlit.

The core of the system revolves around **transfer learning**, where the pre-trained MobileNetV2 model, originally trained on the ImageNet dataset, is adapted for binary classification: distinguishing between "NORMAL" and "PNEUMONIA" chest X-rays. MobileNetV2 is chosen for its balance between computational efficiency and accuracy, making it suitable for deployment even on systems with limited hardware resources. The top layers of the pre-trained model are removed and replaced with a flattening layer, a dense layer with 128 neurons and ReLU activation, and a final softmax layer to output class probabilities.

Upon uploading an image via the Streamlit interface, the image is resized to 224x224 pixels and normalized. The model then processes this image and returns a class prediction along with a **confidence score**, offering a quantitative measure of the model's certainty. If the predicted class is "PNEUMONIA" and the confidence exceeds a defined threshold (e.g., 80%), the system advises a medical consultation.

To address the "black box" nature of deep learning, the system includes a **Grad-CAM heatmap overlay**, which visually highlights the most influential regions in the image that led to the prediction. This overlay is superimposed on the original chest X-ray and presented alongside it on the web interface, allowing healthcare practitioners to visually validate and interpret the model's reasoning.

Moreover, the proposed system's implementation using **Streamlit** offers several benefits: it is lightweight, easy to deploy, and does not require complex backend infrastructure. This enhances accessibility for healthcare providers who may not have extensive technical expertise. The model's architecture, training, and prediction logic are encapsulated in Python scripts, while the GUI is powered by Streamlit and interactive widgets for real-time analysis.

In summary, the proposed system presents an end-to-end framework for **automated pneumonia detection**, combining accuracy, interpretability, and ease of use. By integrating deep learning, visualization, and web technologies, it provides a practical tool that can support radiologists and clinicians in early diagnosis, reduce diagnostic workload, and improve patient outcomes in critical healthcare scenarios.

2. SYSTEM ANALYSIS

3.1 Functional Requirements

Functional requirements are the following:

- **Image Upload**: Allow users to upload chest X-ray images in common formats (e.g., JPEG, PNG) via a user-friendly interface.
- **Image Preprocessing**: Resize images to 224x224 pixels and normalize pixel values for model compatibility.
- **Model Prediction**: Classify images into "NORMAL" and "PNEUMONIA" using the MobileNetV2 model, providing a confidence score for each prediction.
- **Grad-CAM Visualization**: Generate and superimpose a Grad-CAM heatmap on the original image to highlight contributing areas.
- **Result Display**: Clearly display predictions and provide advice on whether to consult a doctor based on the results.
- **Error Handling**: Gracefully handle errors with informative messages for invalid uploads or processing issues.
- **User Documentation**: Include documentation explaining how to use the application and interpret results.
- **Compatibility**: Ensure compatibility with various operating systems (e.g., Windows, macOS, Linux) for broad accessibility.

2.1 Non-Functional Requirements

2.1.1 Performance Requirements

- **Scalability:** The system shall be designed to handle multiple concurrent users without significant degradation in performance.
- **Reliability:** The system shall maintain high availability, ensuring that users can access the application at any time without unexpected downtime.
- **Accuracy:** The model shall achieve a high level of accuracy in predictions, with a target accuracy rate of at least 85% for classifying chest X-ray images.

3.2.2 System Requirements

- **Data Protection:** The system shall implement security measures to protect user data and uploaded images, ensuring that sensitive information is not exposed**.**
- **Access Control:** The system shall enforce access control measures to restrict unauthorized access to the application and its data.

3.2.3 User Interface

This UI design prioritizes ease of use, providing quick access to diagnostic insights with clear visual and textual feedback to support clinical decision-making.

2.2 Software Requirements

1. Operating System: Windows 10
2. Web browser: Chrome, Microsoft Edge, Mozilla Firefox
3. Command prompt
4. Programming language: Python
5. Libraries: tensor flow ,keras ,opencv ,matplotlib

2.3 Hardware Requirements

1. Processor: intel core i3
2. Storage: 10GB (variable)
3. RAM: 8GB
4. Input devices: keyboard and mouse

2.4 Block Diagram Of The Proposed System

Fig 3.5: Block diagram of the system

The figure describes the proposed system for pneumonia detection using chest X-ray images, structured in several sequential stages that work together to automate and enhance the diagnostic process. The system begins with taking a chest X-ray image as input. Users can upload the image through an intuitive graphical user interface (GUI), making the process straightforward for healthcare professionals. This user-friendly design facilitates seamless integration of the diagnostic tool into clinical workflows.

Following image input, the system performs preprocessing to prepare the image for analysis. The chest X-ray is resized to 224 by 224 pixels to ensure consistent input size. Normalization is applied by scaling pixel values between 0 and 1, which helps standardize the data and enhance the model's performance. Additionally, the image is converted into a format compatible with the deep learning model, providing uniformity for effective feature extraction.

At the core of the system is a deep learning model based on MobileNetV2 enhanced with custom layers. Leveraging transfer learning, the model extracts high-level features from the processed X-ray image.

The dense layers appended to MobileNetV2 classify the image into one of two categories: "NORMAL" or "PNEUMONIA." The model generates probability scores for each class, and the system selects the prediction with the highest confidence. If the confidence level for "PNEUMONIA" exceeds a specified threshold commonly set at 80% the system flags the case for medical consultation.

To provide transparency and interpretability, the system generates a Grad-CAM (Gradient-weighted Class Activation Mapping) heatmap that visually highlights regions in the X-ray influencing the model's decision. This heatmap assists clinicians in understanding the reasoning behind the classification, thereby increasing trust in the automated diagnosis.

Finally, the system displays the results within the GUI, presenting the original chest X-ray image along with the Grad-CAM heatmap superimposed on it. The classification result is shown together with the confidence score, accompanied by a recommendation indicating whether the patient should consult a doctor. This visualization and decision support interface ensures that end-users receive clear, actionable insights based on the model's analysis.

3. SYSTEM DESIGN

4.1 Introduction

The system design for the pneumonia detection model is structured to integrate deep learning techniques with intuitive user interaction, enabling efficient and accurate medical image classification. This design outlines the architectural framework that transforms raw chest X-ray images into meaningful diagnostic insights. It combines a lightweight and powerful convolutional neural network, MobileNetV2, with a user-friendly graphical interface and interpretability tools like Grad-CAM. Each component—from data input and preprocessing to model inference and heatmap visualization—plays a vital role in ensuring the system's reliability and transparency. This carefully crafted design supports healthcare professionals in making informed decisions, especially in the early detection and management of pneumonia.

4.2 UML Diagrams

UML, which stands for Unified Modelling Language, is a way to visually represent the architecture, design, and implementation of complex software systems. Looking at code and understanding the system is difficult, instead we can use UML diagrams to understand the system. UML diagrams can be used to explain the components of the software to people who don't have technical knowledge.

It is a standard language for specifying, visualizing, constructing, and documenting the artefacts of the software systems. UML is different from other common programming languages like C++, Java, and COBOL etc. It is pictorial language used to make software blueprints.

Although typically used in software engineering it is a rich language that can be used to model an application structure, behaviour and even business processes. There are 8 UML diagram types to help us model this behaviour.

4.2.1 State Diagram

A **state diagram** is a type of diagram used in software engineering and systems design to represent the different states of an object or system and how it transitions from one state to another based on events or conditions. It visually depicts states (represented as rounded rectangles) and transitions (arrows) triggered by events. State diagrams are especially useful in modeling the behavior of dynamic systems such as user interfaces, protocols, or control systems, and are a key component of **UML (Unified Modeling Language)**.

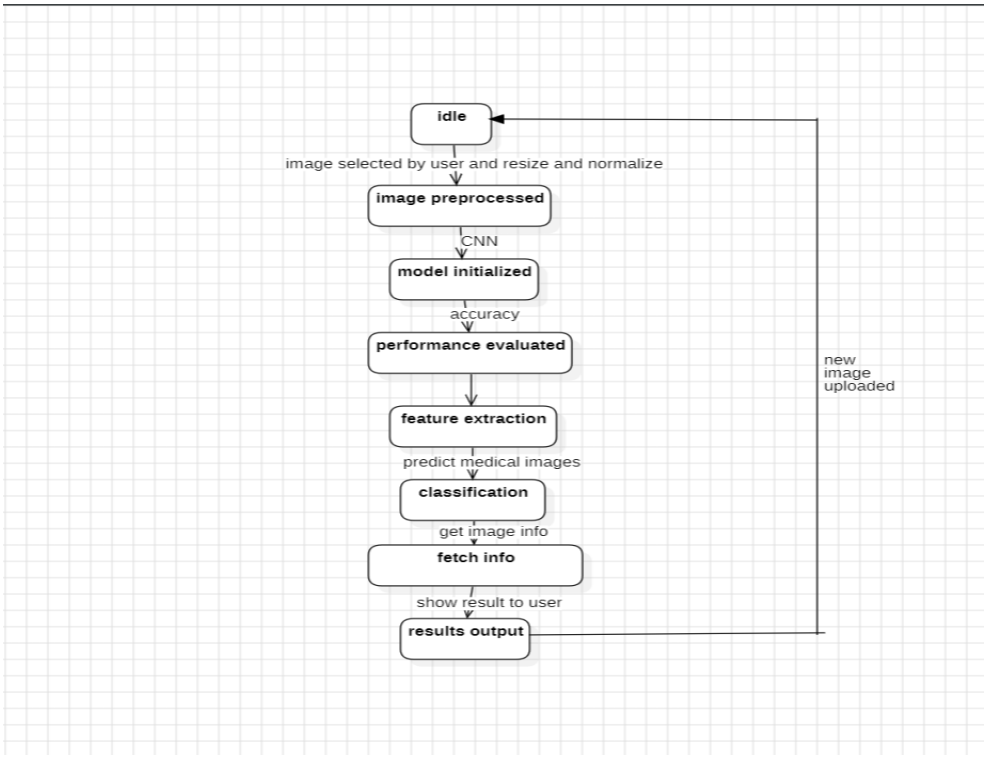


Fig 4.2.1: State diagram

4.2.2 Class Diagram

Because a lot of software is based on object-oriented programming, where developers define types of functions that can be used, class diagrams are the most commonly used type of UML diagram. Class diagrams show the static structure of a system, including classes, their attributes and behaviours, and the relationships between each class. A class is represented by a rectangle that contains three compartments stacked vertically: Class name, attributes and functions of that class with access specifiers. The class diagram also contains association, generalization and aggregation. Where generalization is to show inheritance of a class and aggregation is to show that objects are assembled or configured together to create a more complex object.

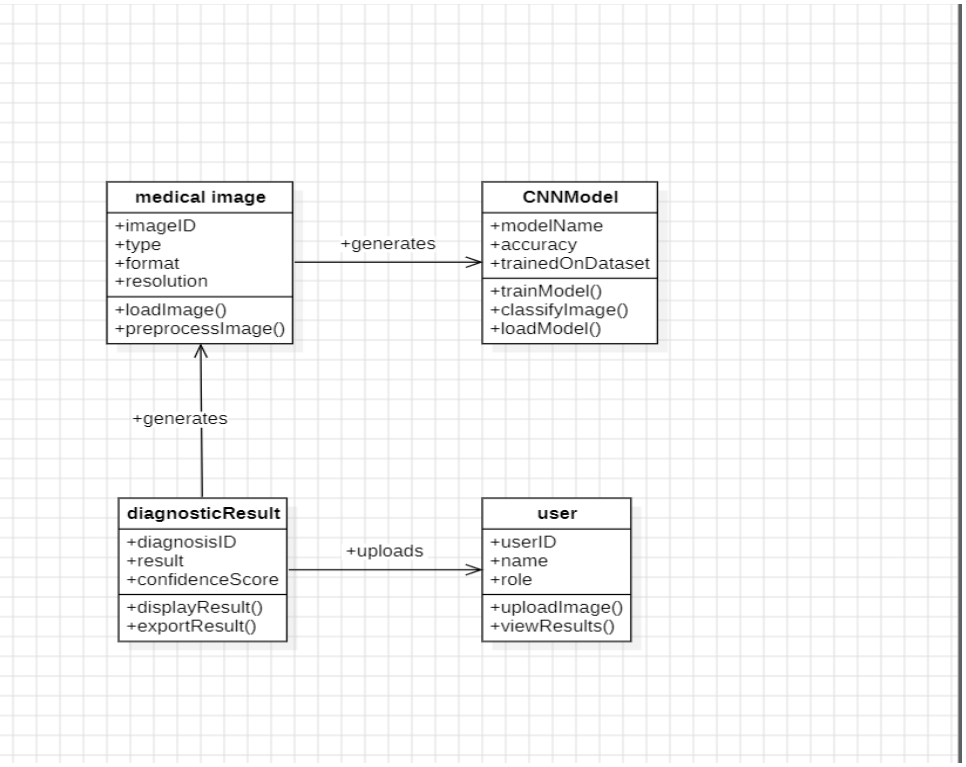


Fig 4.2.2: Class diagram

4.2.3 Activity Diagram

The Unified Modelling Language includes several subsets of diagrams, including structure diagrams, interaction diagrams, and behaviour diagrams. Activity diagrams, along with use case and state machine diagrams, are considered behaviour diagrams because they describe what must happen in the system being modelled. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc

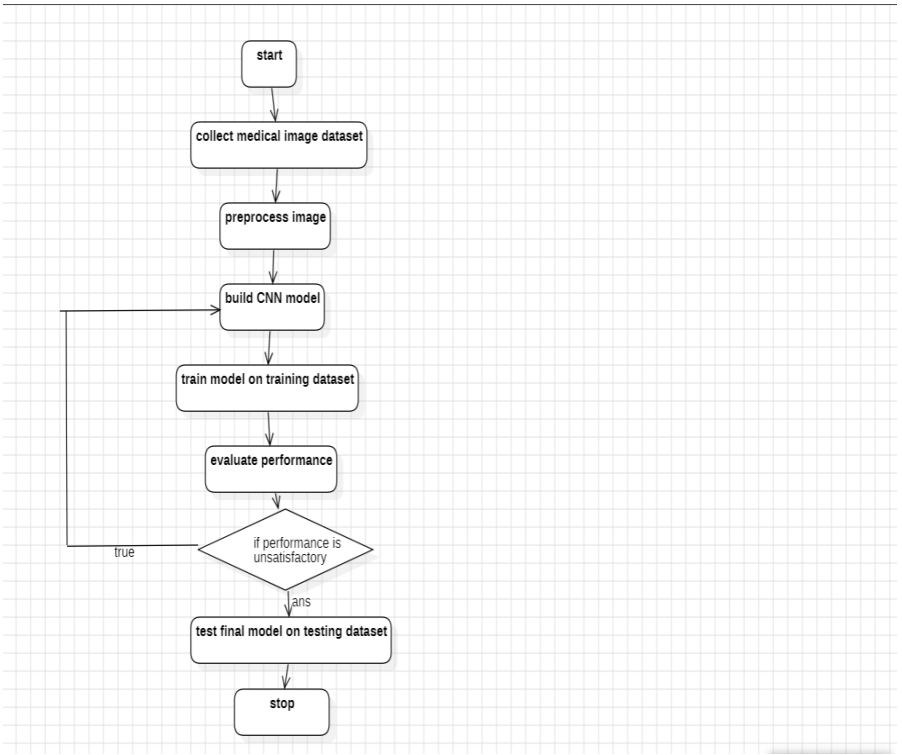


Fig 4.2.3: Activity Diagram

4.2.4 Use case Diagram

A UML use case diagram is the primary form of system/software requirements for a new software program under developed. In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. It is useful in the following situations: Scenarios in which your system or application interacts with people, organizations, or external systems, Goals that your system or application helps those entities (known as actors) achieve, The scope of your system. Use cases specify the expected behaviour (what), and not the exact method of making it happen (how).

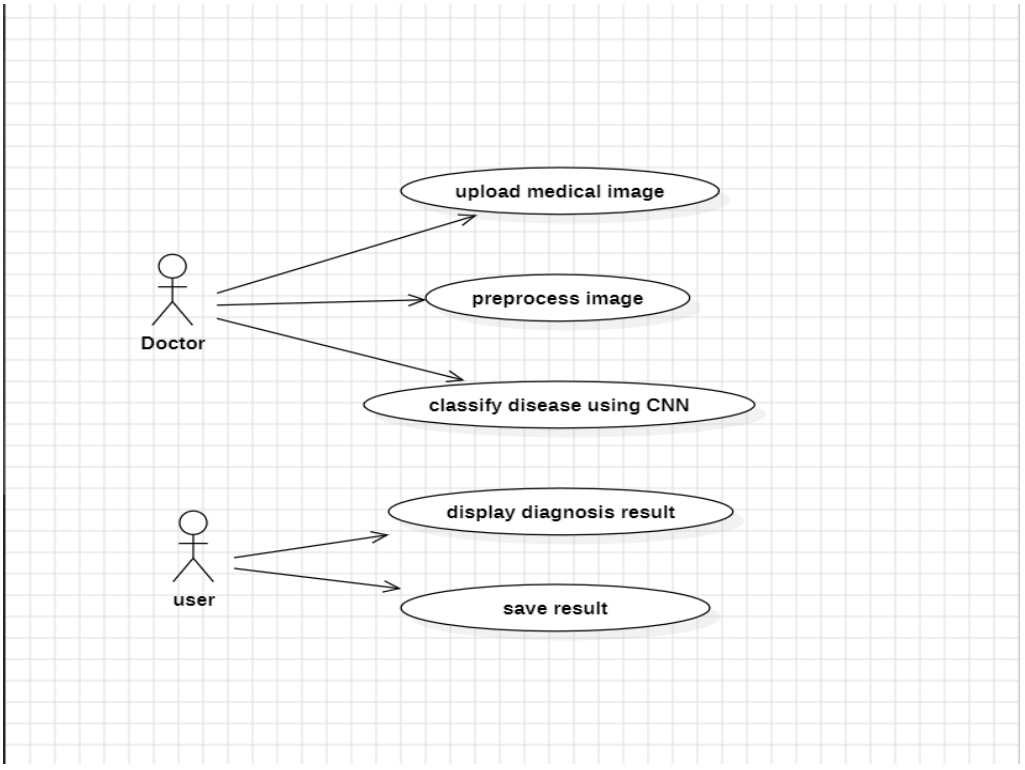


Fig 4.2.4: Use case diagram

4.2.5 Sequence Diagram

UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration, A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

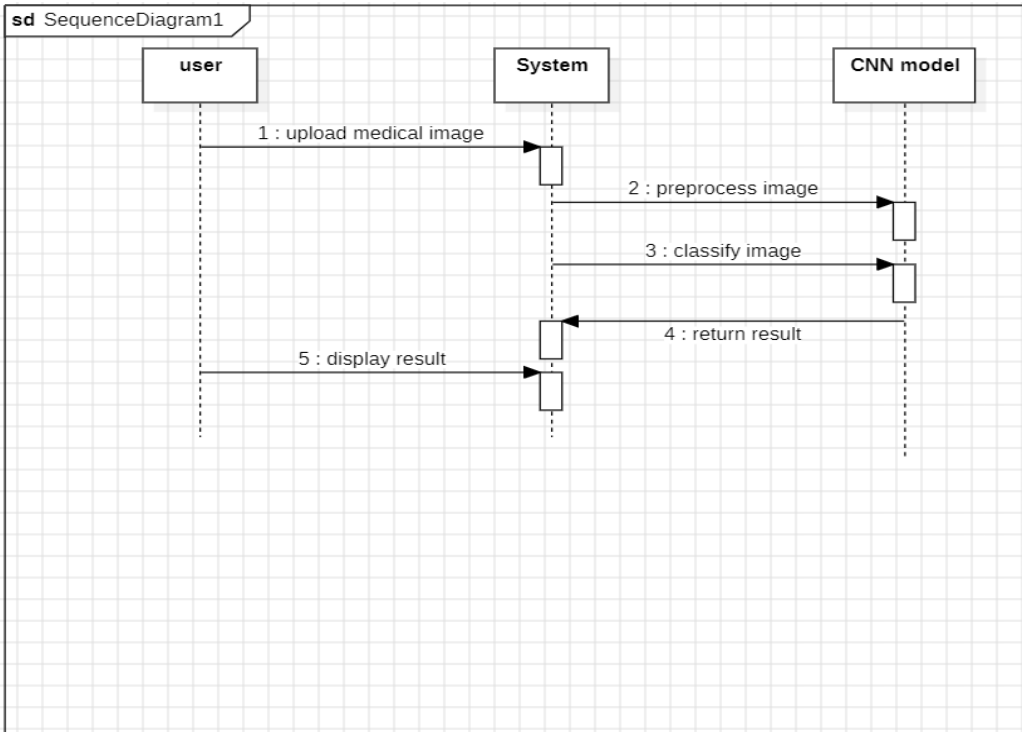


Fig 4.2.5: Sequence diagram

5 IMPLEMENTATION AND RESULTS

5.1 Introduction

The **implementation and results** section outlines the practical development and execution of the proposed medical image classification system. It details how the pneumonia detection model was built using deep learning techniques, specifically leveraging the MobileNetV2 architecture through transfer learning. The implementation includes steps like data preprocessing, model training, validation, and integration with a graphical user interface (GUI) for user interaction.

This section also presents the **experimental results**, showcasing the model's performance using metrics such as accuracy, loss, and confidence levels. Additionally, visual outputs like Grad-CAM heatmaps are used to explain model decisions. The goal is to demonstrate the effectiveness, reliability, and interpretability of the system in accurately detecting pneumonia from chest X-ray images.

5.2 Method of Implementation

In this project, the Waterfall model is effectively followed as the system development methodology. This traditional software development life cycle (SDLC) model is a linear and sequential approach, where each phase flows logically into the next. The stages requirement analysis, system design, implementation, testing, deployment, and maintenance are completed one after another, with little to no overlap.

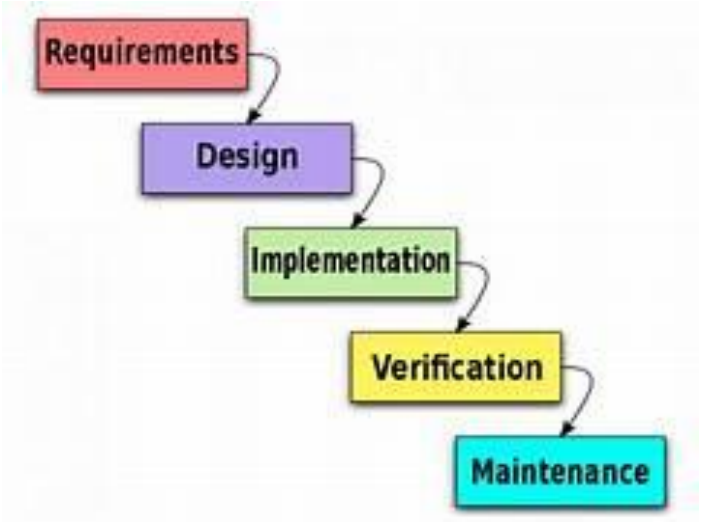


Fig 5.1: WaterFall method

This model suits the project well because the requirements for pneumonia detection using deep learning were clearly defined at the beginning. Once the objective (medical image classification using MobileNetV2 and Grad-CAM visualization) was established, each phase was executed in order from gathering datasets and designing the model architecture, to training, validating, deploying the model, and finally integrating it into a user-friendly GUI. The Waterfall model ensures systematic progress, quality documentation, and easier monitoring of development stages, which is essential for research-based and academic projects like this.

Water fall four main values are:

- Requirement clarity
- Structured Design Phase
- Dedicated Implementation Phase
- Final Testing and Validation

5.2.1 Resize

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os

# ☑ Set the base path where your dataset is located
# CHANGE THIS to the actual folder path if it's different
base_path = "C:/Users/abhis/OneDrive/Desktop/medical-image/chest_xray"

# Data generators with augmentation for training and normalization for all
train_gen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    zoom_range=0.2,
    horizontal_flip=True
)

val_gen = ImageDataGenerator(rescale=1./255)

# ☑ Define full absolute paths to subfolders
train_dir = os.path.join(base_path, 'train')
val_dir = os.path.join(base_path, 'val')
test_dir = os.path.join(base_path, 'test')

# Load training data
train_data = train_gen.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary'
)
```

```
# Load validation data
val_data = val_gen.flow_from_directory(
    val_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary'
)

# Load test data
test_data = val_gen.flow_from_directory(
    test_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',
    shuffle=False
        )
```

5.2.2 Train_model

```
import os
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# --- Configuration ---
img_height = 224
img_width = 224
batch_size = 32
```

```
epochs = 5
num_classes = 2
model_save_path = "pneumonia_model.h5"
weights_save_path = "pneumonia_model.weights.h5"

# --- Data Preparation ---
train_dir = "chest_xray/train"
val_dir = "chest_xray/val"

train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    zoom_range=0.2,
    horizontal_flip=True
)

val_datagen = ImageDataGenerator(rescale=1.0 / 255)

train_gen = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical'
)

val_gen = val_datagen.flow_from_directory(
    val_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical'
)

# --- Model Building ---
base_model = MobileNetV2(weights='imagenet', include_top=False,
```

```
        input_shape=(img_height, img_width, 3))
base_model.trainable = False

x = Flatten()(base_model.output)
x = Dense(128, activation='relu')(x)
output = Dense(num_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=output)

# --- Compile ---
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# --- Train and Record History ---
print("\n Training started...\n")
history = model.fit(train_gen, validation_data=val_gen, epochs=epochs)

# --- Save Model and Weights ---
model.save(model_save_path)
model.save_weights(weights_save_path)

# --- Plot Accuracy and Loss ---
plt.figure(figsize=(12, 5))

# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy', marker='o')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy', marker='o')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# Loss plot
```

```
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss', marker='o')
plt.plot(history.history['val_loss'], label='Validation Loss', marker='o')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.savefig("accuracy_plot.png")
plt.savefig("loss_plot.png")

# --- Final Metrics Output ---
final_train_acc = history.history['accuracy'][-1]
final_val_acc = history.history['val_accuracy'][-1]
final_train_loss = history.history['loss'][-1]
final_val_loss = history.history['val_loss'][-1]

print("\n☑ Final Training Accuracy:", f"{final_train_acc:.4f}")
print("☑ Final Validation Accuracy:", f"{final_val_acc:.4f}")
print("☑ Final Training Loss:", f"{final_train_loss:.4f}")
print("☑ Final Validation Loss:", f"{final_val_loss:.4f}")

# ☑ Paths saved
print("\n☑ Full model saved to:", os.path.abspath(model_save_path))
print("☑ Weights saved to:", os.path.abspath(weights_save_path))
    print("☑ Accuracy and loss plots saved as 'accuracy_plot.png' and
    'loss_plot.png'.")
```

5.2.3 Streamlit_Grad

```
import streamlit as st
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.applications import MobileNetV2
from PIL import Image
import numpy as np
import cv2

# --- Configuration ---
img_height = 224
img_width = 224
num_classes = 2
class_names = ["NORMAL", "PNEUMONIA"]
pneumonia_threshold = 0.8
last_conv_layer_name = "Conv_1"

# --- Load Model ---
base_model = MobileNetV2(weights='imagenet', include_top=False,
input_shape=(img_height, img_width, 3))
base_model.trainable = False

x = Flatten()(base_model.output)
x = Dense(128, activation='relu')(x)
output = Dense(num_classes, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=output)

try:
    model.load_weights("pneumonia_model.weights.h5")
    st.sidebar.success("✅ Model weights loaded successfully.")
except:
```

```
    st.sidebar.warning("⚠️ Trained model weights not found. Using untrained model.")

# --- Grad-CAM Function ---
def make_gradcam_heatmap(img_array, model, base_model, last_conv_layer_name):
    grad_model = Model(inputs=base_model.input,
                       outputs=[base_model.get_layer(last_conv_layer_name).output, model.output])
    with tf.GradientTape() as tape:
        conv_outputs, predictions = grad_model(img_array)
        pred_index = tf.argmax(predictions[0])
        class_channel = predictions[:, pred_index]
    grads = tape.gradient(class_channel, conv_outputs)
    pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2))

    conv_outputs = conv_outputs[0].numpy()
    pooled_grads = pooled_grads.numpy()
    for i in range(pooled_grads.shape[-1]):
        conv_outputs[:, :, i] *= pooled_grads[i]
    heatmap = np.mean(conv_outputs, axis=-1)
    heatmap = np.maximum(heatmap, 0) / np.max(heatmap)
    return heatmap

# --- Streamlit UI ---
st.title("🩺 Medical Image Diagnosis for Pneumonia Detection ")

uploaded_file = st.file_uploader("Upload a Chest X-ray image", type=["jpg", "jpeg", "png"])

if uploaded_file:
    # Load and preprocess image
```

```
    image_pil = Image.open(uploaded_file).convert("RGB")
    image_resized = image_pil.resize((img_width, img_height))
    img_array = tf.keras.utils.img_to_array(image_resized)
    img_array_exp = np.expand_dims(img_array, axis=0)
    img_array_exp =
tf.keras.applications.mobilenet_v2.preprocess_input(img_array_exp)

    # Model prediction
    preds = model.predict(img_array_exp)
    class_index = np.argmax(preds[0])
    confidence = preds[0][class_index] * 100
    label = class_names[class_index]
    advice = "Yes" if label == "PNEUMONIA" and confidence >
pneumonia_threshold * 100 else "No"

    st.markdown(f"<p style='font-size:14px;'>### 💉 Prediction:
<b>{label}</b></p>", unsafe_allow_html=True)
    st.markdown(f"<p style='font-size:14px;'><b>Disease Percentage</b>:
{confidence:.2f}%</p>", unsafe_allow_html=True)
    st.markdown(f"<p style='font-size:14px;'><b>Consult a Doctor</b>:
{advice}</p>", unsafe_allow_html=True)

    # Styled message boxes instead of st.success / st.info
    if advice == "Yes":
        st.markdown(
            """
            <div style='background-color:#cff4fc; border-left:4px solid #0dcaf0;
padding:10px; font-size:14px;'>
            🔔 The model suggests possible signs of pneumonia. Please consult a
medical professional for confirmation.
            </div>
            """,
```

```
            unsafe_allow_html=True
        )
    else:
        st.markdown(
            """
            <div style='background-color:#d1e7dd; border-left:4px solid #198754;
padding:10px; font-size:14px;'>
            ✅ No strong signs of pneumonia detected. If symptoms persist, still
consider clinical evaluation.
            </div>
            """,
            unsafe_allow_html=True
        )

    # Generate Grad-CAM
    heatmap = make_gradcam_heatmap(img_array_exp, model, base_model,
last_conv_layer_name)
    heatmap_resized = cv2.resize(heatmap, (img_width, img_height))
    heatmap_colored = cv2.applyColorMap(np.uint8(255 * heatmap_resized),
cv2.COLORMAP_JET)
    heatmap_colored = cv2.cvtColor(heatmap_colored, cv2.COLOR_BGR2RGB)
    superimposed_img = cv2.addWeighted(np.array(image_resized), 0.6,
heatmap_colored, 0.4, 0)

    # Grad-CAM Explanation ABOVE Image
    st.markdown("---")
    st.subheader("📊 Grad-CAM Heatmap Interpretation")
    st.markdown(
        """
        <div style='font-size:14px;'>
        The Grad-CAM heatmap shows which areas of the chest X-ray were most
important in the model’s prediction.<br><br>
```

```
        - 🔴 <b>Red/Yellow regions</b> highlight areas that strongly influenced the decision (possible infection zones).<br>
        - 🔵 <b>Blue or dark regions</b> indicate less relevance to the model.<br>
        - This helps medical professionals understand the model's focus during classification.
        </div>
        """, unsafe_allow_html=True
    )

    # Show heatmap overlay image
    st.image(superimposed_img, caption="Grad-CAM Heatmap Overlay", use_container_width=True)
```

5.3 Algorithms and Flowcharts

The pneumonia detection system follows a structured algorithm where the process begins by uploading a chest X-ray image through a user-friendly graphical interface. The uploaded image is preprocessed, including resizing to 224x224 pixels and normalizing pixel values, to ensure compatibility with the model. The system then loads the pre-trained MobileNetV2 deep learning model, which has been fine-tuned for classifying chest X-rays into two categories: NORMAL and PNEUMONIA. Once the image is processed by the model, it outputs the predicted class along with a confidence score. To provide visual interpretability, the system uses Grad-CAM (Gradient-weighted Class Activation Mapping) to generate a heatmap that highlights the areas of the image most influential in the model's decision-making. This heatmap is superimposed on the original X-ray and displayed along with the prediction and a recommendation on whether to consult a doctor based on the model's confidence. The logical flow ensures a seamless experience from image input to diagnostic insight, aiding healthcare professionals in timely and accurate pneumonia detection.

5.4 Control Flow of the Implementation

The implementation of the medical image classification system begins with the collection and preprocessing of medical imaging data, such as chest X-rays. These images are first standardized through resizing, normalization, and, in some cases, data augmentation to improve the robustness of the model. The preprocessed images are then split into training, validation, and test sets to ensure a reliable evaluation of model performance. A deep learning model, typically based on convolutional neural networks (CNNs), is constructed or fine-tuned using transfer learning from a pre-trained architecture such as MobileNetV2. The control flow initiates with the input image passing through multiple convolutional and pooling layers, which automatically learn hierarchical features relevant to medical diagnosis. These features are then fed into fully connected layers to perform classification, outputting a probability score indicating the presence or absence of

a condition like pneumonia. The model is trained using backpropagation and an appropriate loss function, with optimization carried out via algorithms such as Adam or SGD. During training, performance metrics such as accuracy, precision, recall, and loss are monitored on the validation set to prevent overfitting. Once trained, the model is evaluated on the test set to confirm its generalization ability. Additionally, techniques like Grad-CAM (Gradient-weighted Class Activation Mapping) can be integrated to visualize which areas of the image the model focused on, enhancing interpretability. The final deployed system can be used as a decision-support tool in clinical settings, where a user inputs an X-ray image, and the system outputs a diagnostic prediction along with a heatmap indicating key regions contributing to the decision.

Importing Dependencies

- **TensorFlow:** Core deep learning library used to build and train the MobileNetV2-based model.
- **Keras:** High-level API (built into TensorFlow) for creating and managing neural networks.
- **Numpy:** Used for handling image arrays and performing mathematical operations on data.
- **Matplotlib:** For visualizing training accuracy/loss curves and Grad-CAM heatmaps.
- **OpenCV:** Used for image processing, including resizing, coloring, and overlaying heatmaps on X-rays.
- **Python:** Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.
- **Pillow:** PIL (Python Imaging Library) is a powerful and easy-to-use library in Python for opening, manipulating, and saving many different image file formats.
- **Streamlit:** Streamlit is an open-source Python framework designed specifically for building interactive web applications for data science and machine learning projects.

5.5 Sample Code

- **Python:** Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.
- **Pretrained Model – MobileNetV2:** MobileNetV2 is a lightweight deep learning model designed for efficient performance on devices with limited resources. It is trained on ImageNet and used here as the base feature extractor.
- **.h5 / .weights.h5 Files:**

  .h5: A file format used to save the entire Keras model, including architecture, weights, and optimizer state.

  .weights.h5: Stores only the trained weights of the model, used to reload weights into a model architecture later.
- **Grad-CAM (Gradient-weighted Class Activation Mapping):** Grad-CAM is a visualization technique that highlights the regions in an image that the model focused on when making a prediction.

5.6 Output Screens

Result analysis is a static analysis that determines which functions in each program can return multiple results in an efficient manner.

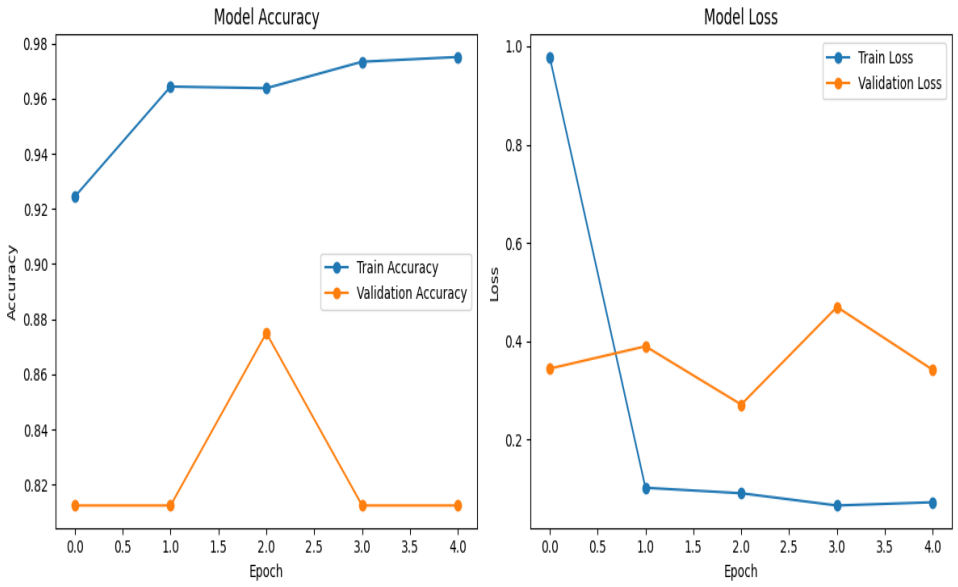


Fig:5.6.1 Output screen of Train_model

This figure shows two side-by-side line charts displaying a machine learning model's performance over five epochs. The left chart illustrates training and validation accuracy, with training accuracy steadily increasing and validation accuracy peaking at epoch 2. The right chart presents training and validation loss, where training loss sharply decreases while validation loss fluctuates. Both charts include labeled axes, grid lines, and legends for clarity. The design is clean, minimal, and fully responsive for different screen sizes.

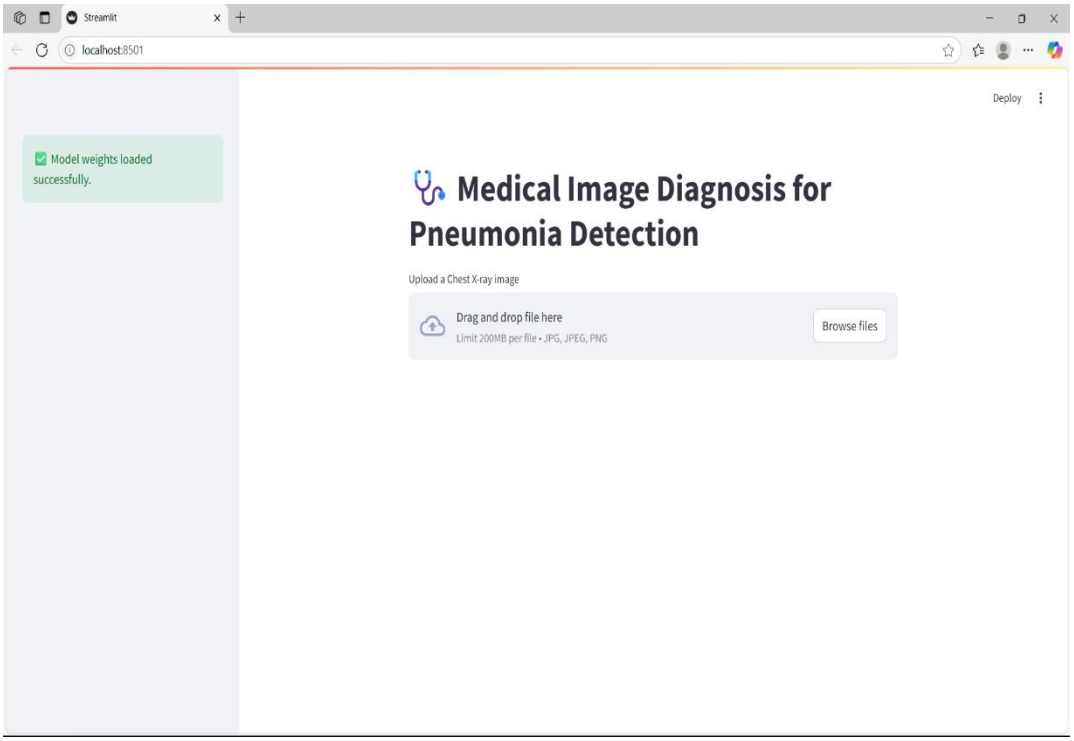


Fig:5.6.2 upload image

The above picture shows the screen to browse files (i.e to upload the image).

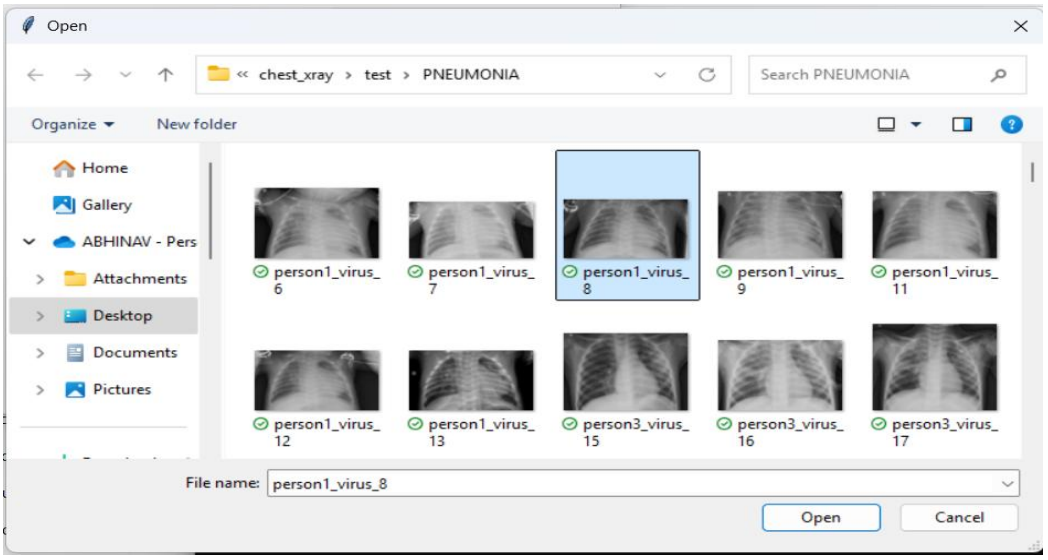


Fig:5.6.3 output screen of pneumonia images

The above figure shows the pneumonia images where we can select any image to get the predicted output.

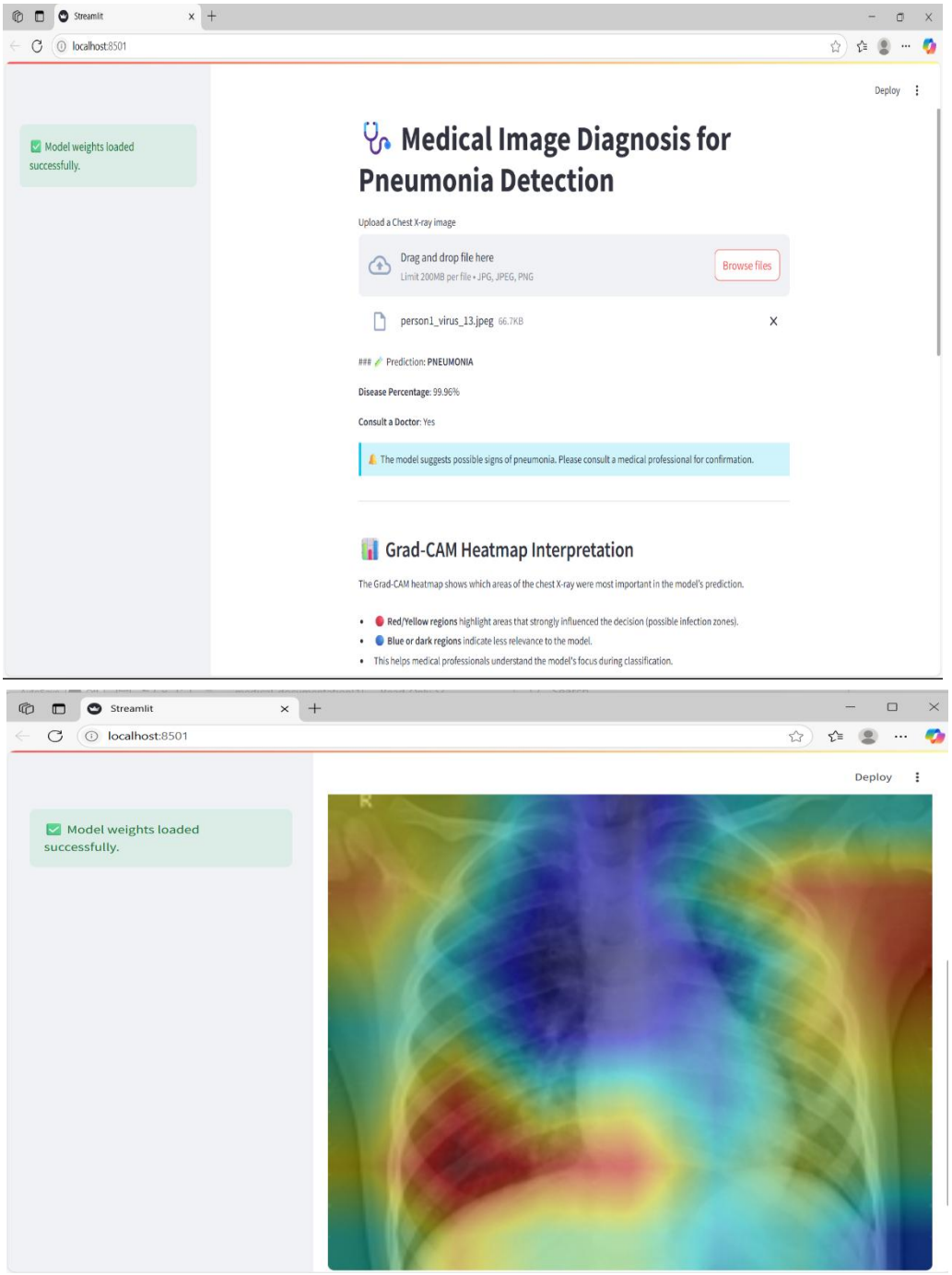


Fig:5.6.4 output screen of pneumonia image

The above figure shows the prediction of pneumonia with the 100% confidence level and also the message whether there is need to consult the doctor or not.

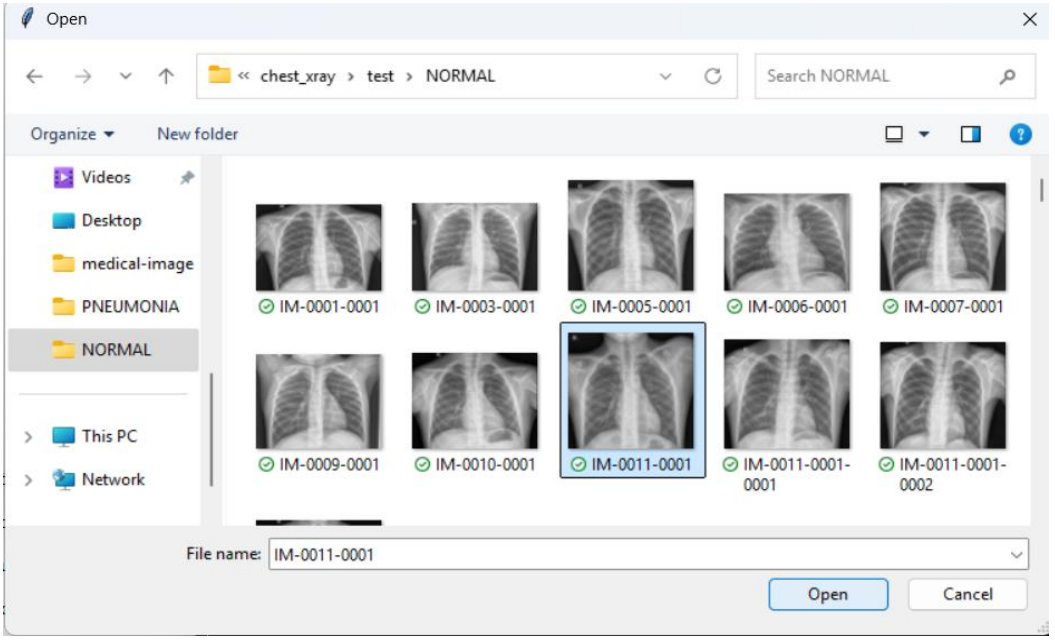
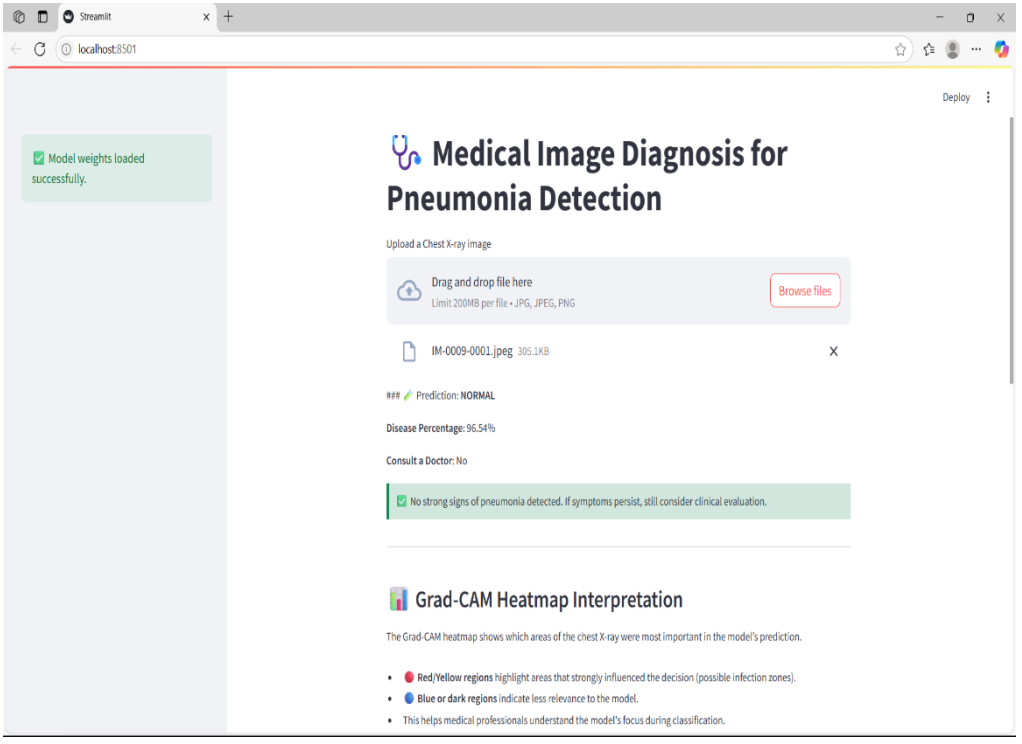


Fig:5.6.5 output screen of normal images

The above figure shows the pneumonia images where we can select any image to get the predicted output.



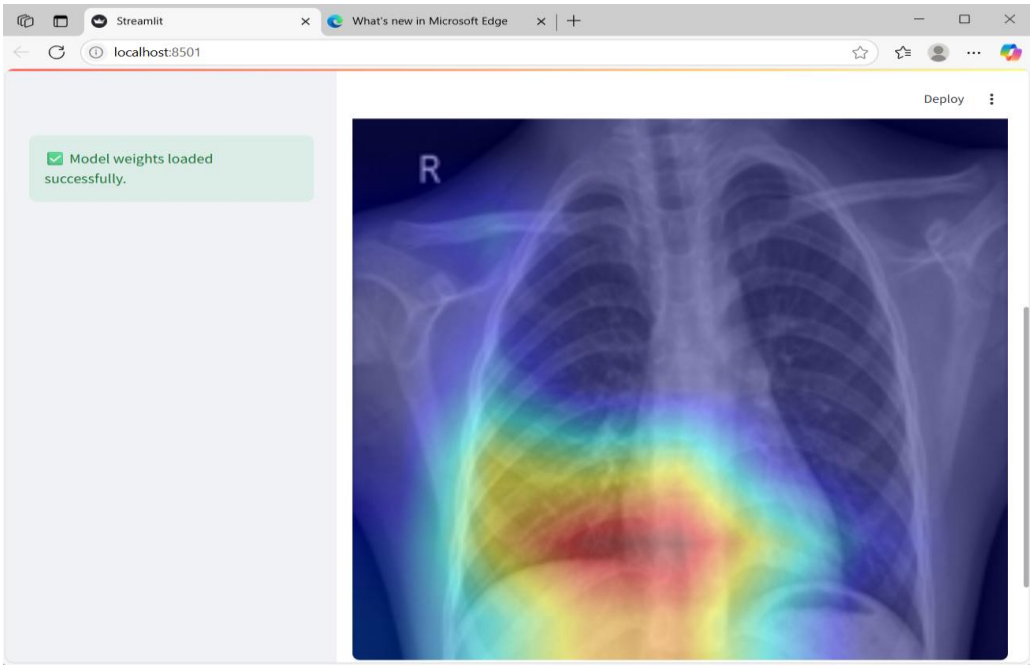


Fig:5.6.6 output screen of normal image

The above figure shows the prediction of pneumonia with the 100% confidence level and also the message whether there is need to consult the doctor or not.

6 TESTING

6.1 Introduction to Testing

Testing is an essential step in the development of the pneumonia detection application, ensuring that it meets the required standards for accuracy, usability, and reliability. By thoroughly testing the model and the application, developers can provide a robust tool that aids healthcare professionals in diagnosing pneumonia from chest X-ray images effectively.

6.2 Types of Tests Considered

- **Unit Testing:**Testing the make grad_cam heatmap function to ensure it correctly generates a heatmap from the model's predictions.
- **Integration Testing**: Testing the integration of the Grad-CAM visualization with the model's output to confirm that the heatmap is generated and displayed correctly.
- **System Testing**: It is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system.
- **Performance Testing**:Evaluating the application's performance with different image sizes and formats to ensure it handles various inputs efficiently.

6.3 Various Test Case Scenarios

Test case writing is a major activity and considered as one of the most important parts of software testing. It is used by the testing team, development team as well as the management. If there is no documentation for an application, we can use the test case as a baseline document.

TEST CASE ID	TEST CASE SCENARIO	INPUTS	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
1	Valid Image Upload	Valid chest X-ray image file	Image processed and displayed in web page	Image displayed in web page	pass
2	Boundary Confidence Score	Image with confidence around 80%	Correct advice returned based on prediction	Correct advise was returned	pass
3	Normal Prediction	Known "NORMAL" chest X-ray image	Model prediction: "NORMAL" with confidence	Predicted the normal with confidence	pass
4	Pneumonia Prediction	Known pneumonia X-ray image	Model prediction: "PNEUMONIA" with confidence	Predicted the pneumonia with confidence	pass
5	Result Display	Prediction results	Prediction and advice displayed accurately	Result was displayed correctly	pass

Fig:6.3 test cases for proposed systems

7 CONCLUSION & FUTURE ENHANCEMENT

7.1 Project Conclusion

The pneumonia detection application developed using deep learning techniques demonstrates a significant advancement in the field of medical imaging analysis. By leveraging convolutional neural networks (CNNs) and Grad-CAM visualization, the application effectively classifies chest X-ray images as either "NORMAL" or "PNEUMONIA." The model's ability to provide visual explanations for its predictions enhances the interpretability of the results, which is crucial in a clinical setting where healthcare professionals rely on clear insights to make informed decisions. Overall, this project contributes to the ongoing efforts to improve diagnostic tools in healthcare, potentially aiding in the early detection and treatment of pneumonia.

7.2 Future Enhancement

Future enhancements for the pneumonia detection application can significantly improve its capabilities and user experience. Key improvements include exploring additional pre-trained models like ResNet or DenseNet to enhance classification accuracy, implementing data augmentation techniques to diversify the training dataset, and expanding the application to support multi-class classification for other lung diseases. Enhancing the user interface to be more intuitive, incorporating features like drag-and-drop image upload, and developing a mobile version would increase accessibility for healthcare professionals. Additionally, integrating the application with electronic health records (EHR) systems would streamline workflows, while a user feedback mechanism would facilitate continuous refinement. Finally, optimizing the model for faster inference times is crucial to ensure quick predictions in clinical settings. By pursuing these enhancements, the application can evolve into a more powerful tool that effectively supports timely and accurate patient care.

8 REFERENCES

8.1 Journals

1. Zhou et al. (2021) authored the paper *"Multi-Modal Broad Learning System for Medical Image and Text-Based Classification,"* presented at the 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), held virtually from October 31 to November 4, 2021.

2. Lin Zhang, Zenglin Qiao, and Lina Li (2024) wrote the article *"An Evolutionary Deep Learning Method Based on Improved Heap-Based Optimization for Medical Image Classification and Diagnosis,"* published in IEEE Access on July 25, 2024.

3. Ao Zhang, Zhenghua Guan, Tengda Zhang, Wenzheng Hu, Yi Liu, and Baiying Lei (2025) presented the paper *"DARNet: A Dual Attention Residual Network for Medical Image Classification"* at the 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

4. The paper titled "Medical Image Recognition Using a Novel Neural Network Construction Controlled by a Kernel Method Encoder" was authored by PENGZHI LI, YAN PEI, JIANQIANG LI, and HAIHUA XIE, and published in the year 2024.

5. The paper titled "Application of Semi-Supervised Learning in Image Classification: Research on Fusion of Labeled and Unlabeled Data" was authored by Sai Li, Peng Kou, Miao Ma, Haoyu Yang, Shuo Huang, and Zhengyi Yang, and published in the year 2024.

8.2 Books

- Python: Python Programming for Beginners by Adam Stark

8.3 Sites

- https://ieeexplore.ieee.org
- www.geeksforgeeks.org