Development of an Android based Pneumonia detection system using Deep Learning

A Project Report Submitted in Partial Fulfillment of Requirements for the Degree of

Bachelor of Technology in Information Technology
(Batch: 2020-2024)

Submitted by
Uddhav Gupta (20BTechIT01)
Amisha Sharma (20BTechIT22)
Siddhant Das (20BTechIT45)
Manab Kumar Das (20BTechIT37)

Under the supervision of:
Prof. Md. Iftekhar Hussain

Department of Information Technology
School of Technology
North-Eastern Hill University, Shillong

Abstract

The proposed Android app leverages deep learning models to classify chest X-ray images into normal and pneumonia categories, emphasizing the crucial role of X-ray imaging in medical diagnostics. X-rays are vital for diagnosing various medical conditions, including pneumonia, due to their ability to provide clear images of the internal structures of the chest. Accurate classification of X-ray images can significantly enhance diagnostic processes, leading to timely and effective treatments. Utilizing neural networks like EfficientNetB7, VGG16, ResNet50v2, and EfficientNetB3 through transfer learning, the app offers robust and accurate image classification. This application aims to assist healthcare practitioners in making preliminary diagnoses of pneumonia, providing a valuable second opinion based on advanced image processing.

User authentication is an essential component of the app, ensuring secure access via sign-up, login, and logout options. Once verified, users can effortlessly upload chest X-ray images from their devices. The application then examines the photos and makes predictions about whether the X-ray reveals signs of pneumonia or is normal. Each forecast is accompanied by a confidence rating, which allows users to quantify the certainty of the results.

By combining deep learning models with user-friendly features and a large dataset, this application seeks to improve diagnostic accuracy and efficiency in healthcare settings. It provides a simple and safe experience for healthcare workers looking to use ML-powered solutions in the fight against pneumonia, ultimately leading to better patient outcomes and more efficient medical workflows.

Acknowledgements

We are grateful to **Prof. Md. Iftekhar Hussain, our project guide and Dean, School of Technology, NEHU**, for his important time and direction during the project, which eventually led us to a beneficial result. His continual mentorship and motivation instilled enthusiasm in us, allowing us to complete the major project on schedule.

We are grateful to **Prof. Debdatta Kandar, Head, Department of Information Technology, NEHU**, for giving us all of the department's resources for carrying out this major project. We would like to thank all of the faculty and staff members of the IT Department for providing us with the necessary facilities and assistance in completing our major project.

We are incredibly grateful to **our parents** for their ongoing support and encouragement, as well as to our friends and well-wishers for their assistance, cooperation, and problem-solving solutions during our major project.

Declaration

This is to certify that we have correctly cited research papers and code documentation collected from various sources and that we have obtained permission for any copyrights required in this project report. We accept all responsibility for the code supplied as part of this major project, as well as the content of this project report.

Uddhav Gupta (20BTechIT01)

Amisha Sharma (20BTechIT22)

Siddhant Das (20BTechIT45)

Manab Kumar Das (20BTechIT37)

Certificate

This is to certify that **Uddhav Gupta** (20BTechIT01), **Amisha Sharma** (20BTechIT22), **Siddhant Das** (20BTechIT45), and **Manab Kumar Das** (20BTechIT37) worked on the major project **Development of an Android based Pneumonia detection system using Deep Learning** from February 2024 to June 2024 and has successfully completed the major project, in order to partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology under my supervision and guidance.

Prof. Md. Iftekhar Hussain
Professor
Department of Information Technology
North-Eastern Hill University
Shillong-793022, Meghalaya, India

Certificate

This is to certify that **Uddhav Gupta** (20BTechIT01), **Amisha Sharma** (20BTechIT22), **Siddhant Das** (20BTechIT45), and **Manab Kumar Das** (20BTechIT37) worked on the major project **Development of an Android based Pneumonia detection system using Deep Learning** from February 2024 to June 2024 and has successfully completed the major project, in order to partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology.

External Examiner

Head of Department

Department of Information Technology
North-Eastern Hill University
Shillong-793022, Meghalaya, India

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 X-Ray image classification

X-ray image classification is a procedure that uses deep learning models, to analyse and categorise X-ray images. These models are trained on large labelled datasets to identify patterns and traits associated with specific circumstances, such as normal or diseased states. During this training, the models learn to distinguish between healthy and diseased tissues, allowing them to make correct predictions when new X-ray pictures are fed in. This procedure includes a number of preprocessing stages, such as image scaling, normalisation, and augmentation, to guarantee that the models generalise well across diverse data sets.

In the medical field, X-ray classification of images is an important diagnostic tool for healthcare practitioners, providing quick and precise predictions for diseases such as pneumonia. This technology improves diagnostic accuracy and efficiency, allowing clinicians to make more informed judgements while lowering the probability of human mistakes.

1.2 Transfer Learning for X-Ray image classification

Transfer learning is a strong deep learning technique in which a pre-trained model that was previously trained on a huge dataset is fine-tuned for specific tasks. In the context of identifying chest X-ray pictures, transfer learning employs models that include EfficientNetB7, VGG16, ResNet50V2, and EfficientNetB3, which have been pre-trained on large datasets like ImageNet. These models have already trained to recognise many elements and patterns in images, laying a solid foundation for specialised tasks.

These pre-trained models are fine-tuned using a large, labelled dataset chest X-ray images to classify them into normal and pneumonia categories. The pre-trained model's final classification layer is replaced with a new layer suited to the specific categories (normal and pneumonia), and the model is then trained on the new dataset. Data preprocessing techniques such as image scaling, normalisation, and augmentation are used to ensure that the model generalises successfully across a wide range of datasets.

This approach significantly reduces the time and computational resources needed compared to training a model from scratch. Transfer learning improves diagnostic accuracy and efficiency, enabling healthcare practitioners to make more informed decisions and providing a reliable second opinion in diagnosing pneumonia from chest X-rays.

1.3 Model Architecture Overview

The proposed model uses EfficientNetB7 with transfer learning to classify chest X-ray images into normal and pneumonia categories.

1. **Base Model Selection:** EfficientNetB7, a trained model trained on ImageNet, was chosen because of its outstanding feature extraction capabilities. The model is set up without the top layers, and global max pooling is used to capture the most important features.

2. **Additional Layers for Classification:** Following the base model, BatchNormalization is used to stabilise and speed up training. A dense layer with softmax activation is incorporated for multi-class classification, allowing the model to efficiently classify chest X-ray pictures as normal or pneumonia.

3. **Compilation and Optimization:** The model is compiled using the Adamax optimizer and fine-tuned with a learning rate of 0.0001. Categorical cross-entropy is used as the loss function, ensuring that the model can minimise prediction errors. The measures used for evaluation include accuracy, precision, and recall, which provide detailed information about the model's performance.

4. **Fine-Tuning Strategy:** A fine-tuning method is used to adapt the previously trained EfficientNetB7 to detect pneumonia. Initially, all EfficientNetB7 layers are frozen to preserve the learned characteristics. As a result, selectable layers beginning with index 100 are unfrozen for further training.

This method helps the model adapt to the specific characteristics of chest X-ray images, improving its performance and diagnostic accuracy.

1.4 Problem Statement

Chest X-ray analysis is critical for identifying respiratory disorders like pneumonia. However, healthcare practitioners must manually evaluate X-ray pictures, which can be time-consuming and error-prone. The lack of specialised expertise in rural or resource-constrained places exacerbates the problem. To solve these challenges, the project plans to create an Android-based pneumonia detection system that employs deep learning algorithms.

The primary objective is to develop an efficient and accurate technology that will help healthcare practitioners identify pneumonia from chest X-ray pictures. The system will use cutting-edge deep learning models, such as EfficientNetB7, VGG16, ResNet50V2, and EfficientNetB3, to analyse X-ray pictures and determine whether they are normal or indicate pneumonia. The project's goal is to provide healthcare practitioners with a trustworthy second opinion by integrating these models into a user-friendly Android application, hence enhancing diagnosis accuracy and efficiency.

Furthermore, the system will incorporate user authentication features to ensure secure access to patient data and comply with privacy regulations. The project aims to address the pressing need for accessible and reliable pneumonia diagnosis tools, particularly in underserved healthcare settings, ultimately contributing to improved patient outcomes and more effective medical workflows.

1.5 Proposed work

The project proposes to develop an Android-based pneumonia detection system using deep learning models and user-friendly features. The key components include:

1. **Deep Learning Model Implementation:** Implementing state-of-the-art deep learning models such as EfficientNetB7, VGG16, ResNet50V2, and EfficientNetB3 for accurate chest X-ray image classification.

2. **User Authentication and Data Security:** Incorporating user authentication features to ensure secure access to patient data and compliance with privacy regulations, enhancing the system's reliability and confidentiality.

3. **Android Application Development:** Designing and developing an intuitive and easy-to-use Android application interface for healthcare practitioners to upload X-ray images, view classification results, and access diagnostic information.

4. **Testing and Validation:** Rigorous testing and validation of the system using x-ray images taken from amicare hospital to ensure robustness, accuracy, and generalization capabilities across different X-ray image variations and patient demographics.

1.6 Overview of Dataset

The chest X-ray image dataset utilised for training has a wide range of images, including both normal and pneumonia cases, all precisely labelled for supervised learning. With over 5500 images, it provides a complete representation of chest X-ray variations, imaging methodologies, and illness presentations. Preprocessing techniques such as scaling, normalisation, and augmentation can improve dataset quality and variability, allowing for more robust model training.

The testing dataset obtained from **Amicare Hospital** extends the validation data by including a subset of chest X-ray pictures gathered in real-world clinical settings. Each image in the collection, which includes both normal and pneumonia cases, is labelled for classification, allowing for a rigorous evaluation of model performance in real-world circumstances. The testing dataset allows for a thorough examination of classification performance and diagnostic accuracy utilising metrics such as accuracy, precision, recall, and F1-score, ultimately contributing to advances in healthcare diagnostics and patient care.

1.7 Organization of the report

The report is organized into six chapters, each covering different aspects of the system. Chapter 1 provides an introduction to the system, setting the stage for the subsequent discussions. Chapter 2 delves into the background study, exploring the foundational concepts and models relevant to the system. Chapter 3 reviews existing work and literature, offering a comprehensive overview of related research. Chapter 4 focuses on the system's design and implementation, detailing the methodology and technical aspects. Chapter 5 presents the results and analyzes the system's performance, highlighting key findings. Finally, Chapter 6 concludes the report, addressing the system's limitations and suggesting directions for future work.

Chapter 2

Backgroud Study

2.1 Classification of X-Ray Image(Anterior- posterior)

X-ray image classification is a specialised application of computer vision and machine learning techniques used to analyse and classify X-ray pictures for diagnostic reasons. X-ray images, often known as radiographs, are commonly employed in medical imaging to reveal internal body structures, notably bones and soft tissues. X-ray classification automates the process of evaluating images to detect certain conditions or abnormalities, such as fractures, tumours, or infections.

Traditionally, X-ray interpretation has relied on radiologists and other healthcare professionals to visually inspect pictures and make diagnostic choices. However, this manual approach can be time-consuming, subjective, and error-prone, especially when dealing with minor or complex irregularities. X-ray image classification seeks to address these issues by using advanced machine learning methods to automatically analyse and interpret X-ray pictures.

X-ray image classification consists of several important phases. First, the X-ray image is preprocessed to improve its quality and reduce noise. Next, features are retrieved from the image using techniques like convolutional neural networks (CNNs), which are specifically designed to detect spatial patterns in images. These features are then sent into a classification algorithm, which places the image in one or more predetermined groups depending on the presence or absence of specific traits.

X-ray image classification has a wide range of applications in healthcare, including screening for diseases including pneumonia, tuberculosis, and lung cancer, as well as detecting fractures and other musculoskeletal disorders. By automating the interpretation of X-ray images, classification algorithms can assist healthcare

workers in making faster, more accurate diagnostic judgements, resulting in better patient outcomes and more efficient healthcare delivery.

2.2 Neural Network

Neural networks are our prevalent model of the brain because they are based on the way neurons are coupled to one another to form networks. Moving the hand to pick up this pencil is an example of a simple piece of information that passes through many processes before becoming a tangible object.

A complete neural network operates simply: The variables are entered as inputs (for instance, an image if the network's purpose is to identify objects in images), and following a series of calculations, an output is produced (using the first example again, an image of a cat should yield the word "cat").

Artificial neural networks are often arranged in columns, with each column's neurons only being able to link to those in columns n+1 and n-1. While there are a few network types that employ distinct architectures.

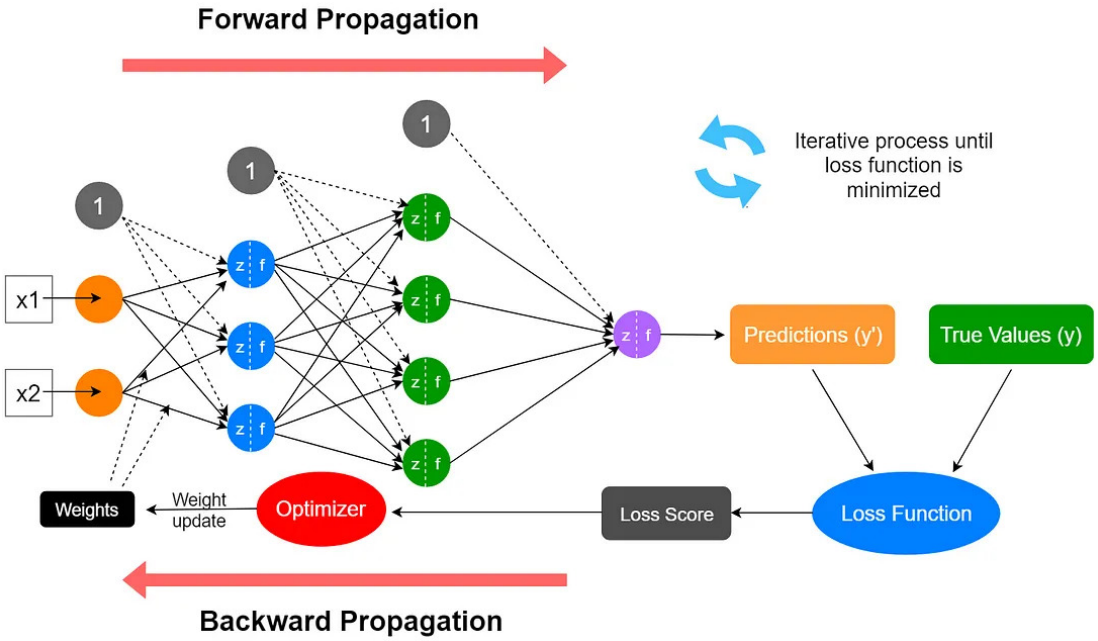

Figure 2.1: Overview of a neural network

2.3 Working of Neuron

The output of the neuron is calculated as follows:

1. Set a threshold value:
    - Threshold = activation value
2. Multiply all inputs with their weights:
    - zi = xi * wi
3. Sum all the results:
    - ws = $\sum_{i=1}^{n} zi + 1 * b$
4. Activation function for Output:
    - Return 1 if the ws $\geq$ Threshold else return 0

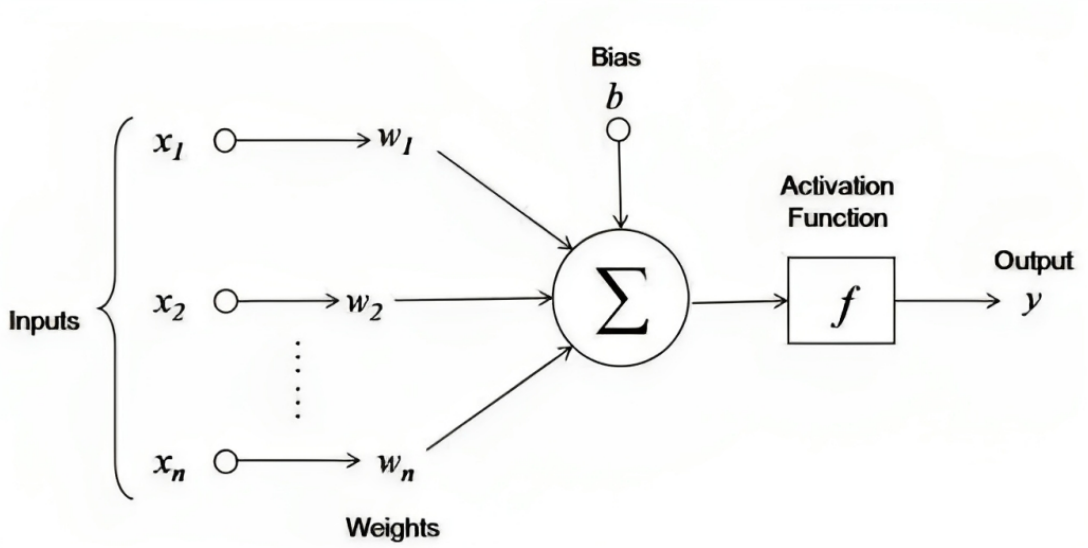

Figure 2.2: Overview of a neural network

2.4 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a deep learning model specifically built for image processing applications. It is made up of layers such as convolutional layers, which use filters on the input image to detect features, activation layers (such as ReLU) to introduce non-linearity, and pooling layers (such as max pooling) to minimise spatial dimensions and computation. These layers extract and downsample characteristics to provide a hierarchical understanding of the image. The last layers are fully integrated and separate the features into distinct categories. CNNs excel at tasks such as image recognition and classification because of their capacity to learn spatial hierarchies and patterns efficiently.

Structure of a Convolutional Neural Network

A Convolutional Neural Network (CNN) is designed to process and classify visual data. The typical structure of a CNN consists of several layers, each serving a specific purpose in extracting and processing features from the input image. Here's a detailed breakdown of the CNN structure:

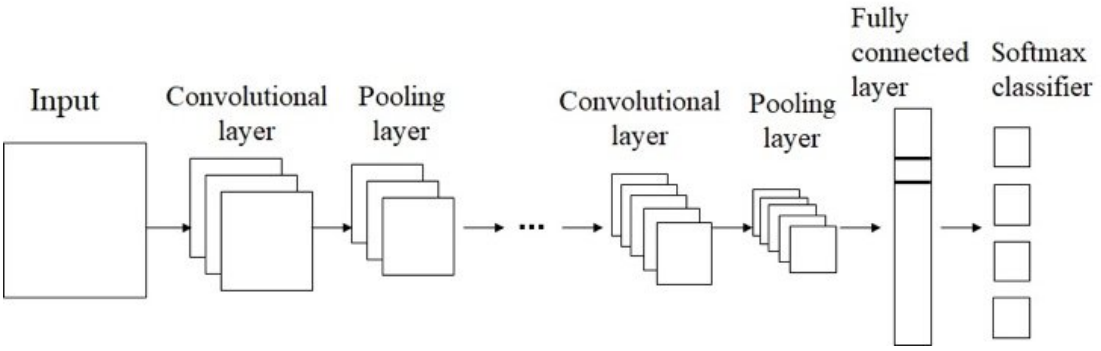

Figure 2.3: Overview of CNN

1. **Input Layer:**
   (a) **Function:** Accepts the raw pixel values of the input image.
   (b) **Shape:** Defined by the dimensions of the input image (e.g., 224x224x3 for a 224x224 RGB image).
2. **Convolutional Layers:**
   (a) **Function:** Apply convolution operations using filters to detect features such as edges, textures, and patterns.

(b) **Components:**

i. **Filters/Kernels:** Small matrices (e.g., 3x3, 5x5) that slide over the input to produce feature maps.

ii. **Stride:** The step size by which the filter moves across the input.

iii. **Padding:** Adds zeros around the input to maintain spatial dimensions.

(c) **Output:** A set of feature maps highlighting the detected features.

3. **Pooling Layers:**

(a) **Function:** Downsample the spatial dimensions of the feature maps, reducing the number of parameters and computational load, and controlling overfitting.

(b) **Types:**

i. **Max Pooling:** Takes the maximum value from each patch of the feature map.

ii. **Average Pooling:** Takes the average value from each patch.

(c) **Output:** Smaller feature maps retaining the most important features

4. **Fully Connected (Dense) Layers:**

(a) **Function:** Flatten the feature maps into a single vector and perform classification based on these features.

(b) **Structure:** Neurons in these layers are fully connected to all activations from the previous layer.

5. **Output Layer:**

(a) **Function:** Provide the final classification result.

(b) **Components:**

i. **Neurons:** The number of neurons corresponds to the number of classes in the classification task.

ii. **Activation Function:**

A. **Softmax:** Used for multi-class classification to output a probability distribution over classes.

B. **Sigmoid:** Used for binary classification.

2.5 Transfer Learning for CNN Models

Transfer learning in CNN models is using a pre-trained network that has previously learnt features from a big dataset to tackle a new but related problem. Transfer learning, rather than starting the training process from scratch, initialise a model with weights from a previously trained model.

The typical process includes:

1. **Selecting a Pre-Trained Model:** Choosing a model like VGG16, ResNet, or EfficientNet pre-trained on a large and diverse dataset.

2. **Adapting the Model:** Modifying the final layers to match the new task's specific requirements, such as changing the output layer to classify a different number of categories.

3. **Fine-Tuning:** Training the model on the new dataset. Often, lower layers are frozen to retain learned features, while higher layers are fine-tuned to adapt to the new data.

This strategy greatly reduces the computational resources and data required, increases performance, and speeds up the training process, making it ideal for applications such as medical picture classification, where labelled data is often limited.

2.6 Transfer learning in x-ray image classification problem

Transfer learning provides significant advantages for X-ray image classification by utilising pre-trained models developed on huge datasets such as ImageNet. It enables the transfer of information gained from big datasets to tasks with little labelled data, hence decreasing the demand for large annotated datasets and computing resources. This method speeds up model training and increases classification accuracy, especially in medical imaging, where labelled data is limited and model training can be resource-intensive. Furthermore, transfer learning allows pre-trained models to be fine-tuned to specific medical imaging applications, improving their performance and generalizability for accurate disease diagnosis from X-ray pictures.

2.7 Convolutional Neural Network Models

EfficientNet-B7

EfficientNet-B7 is part of the EfficientNet family of convolutional neural networks (CNNs) designed for image classification tasks. Developed by researchers at Google, EfficientNet models are known for their ability to achieve high accuracy while being computationally efficient. The key innovation behind EfficientNet is a novel compound scaling method that uniformly scales network width, depth, and resolution.
The model architecture is as follows:

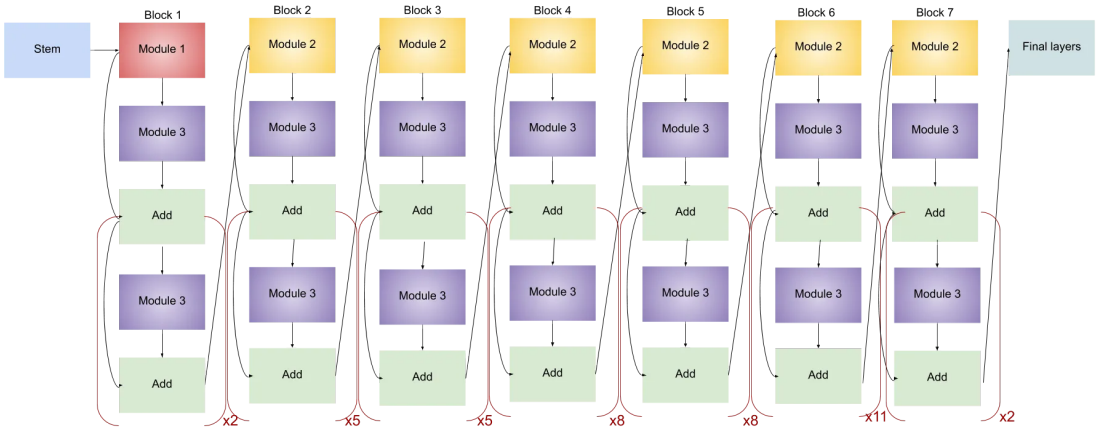

Figure 2.4: EfficientNetB7 Architecture

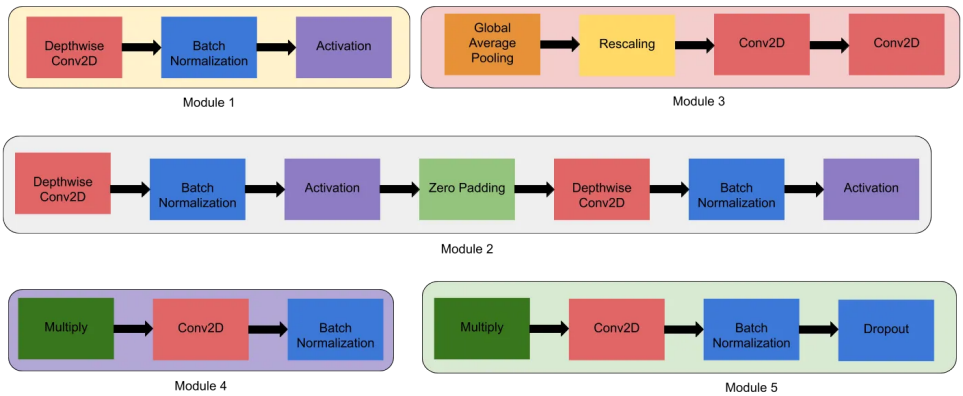

Figure 2.5: Module architecture

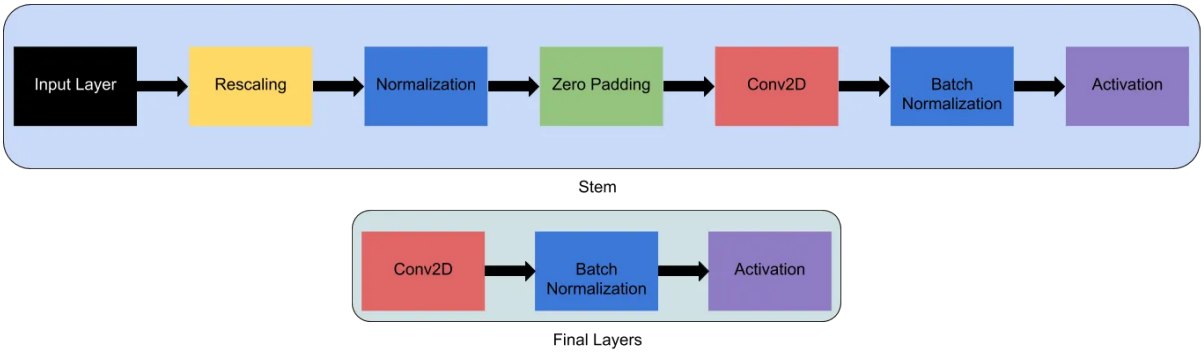

Figure 2.6: Stem and final layer architecture

VGG16

The VGG16 model, developed by the Visual Geometry Group at the University of Oxford, is one of the most widely used convolutional neural network (CNN) architectures. It is known for its simplicity and depth, having 16 weight layers.

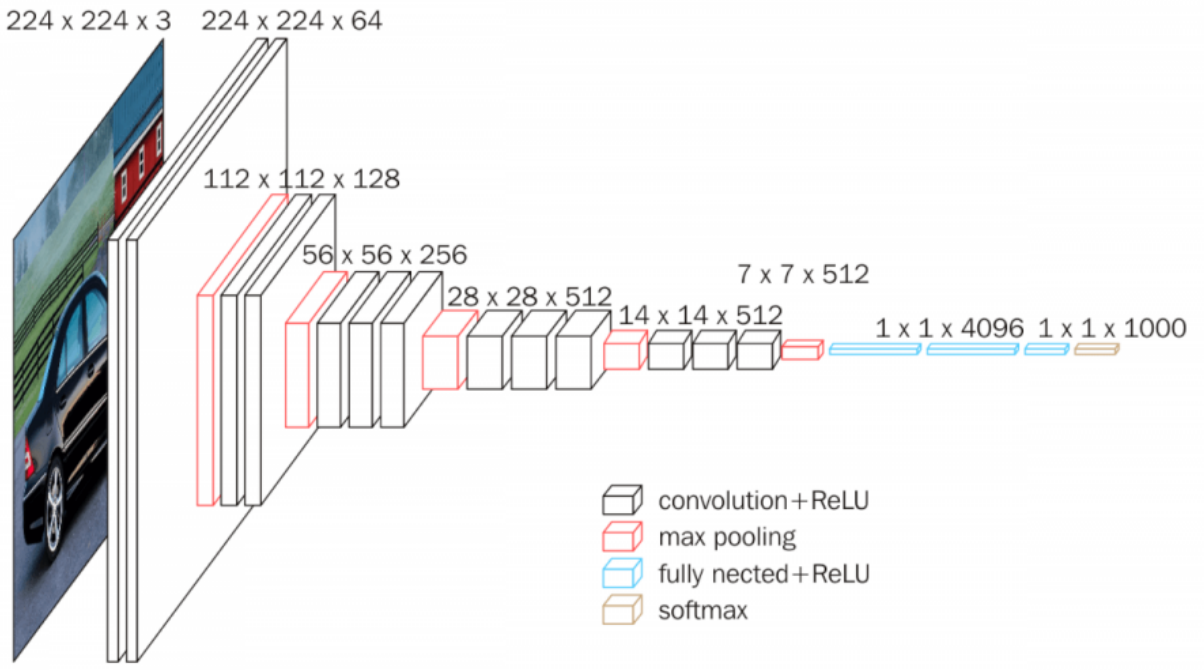

Figure 2.7: VGG16 architecture

The VGG16 architecture consists of the following key components:

1. **Convolutional Layers**: These layers apply a set of filters to the input image, which helps in extracting features such as edges, textures, and patterns.
2. **Max-Pooling Layers**: These layers reduce the spatial dimensions (height and width) of the feature maps while retaining the most important information.
3. **Fully Connected (Dense) Layers**: These layers are used for classification. They take the high-level features extracted by the convolutional layers and output the class scores.
4. **Activation Functions**: The ReLU (Rectified Linear Unit) activation function is used to introduce non-linearity into the model, helping it learn complex patterns.
5. **Softmax Layer**: The final layer uses the softmax activation function to output probabilities for each class.

Detailed Layer-by-Layer Description:

1. **Input Layer:**
    - The input to the VGG16 model is a fixed-size 224x224 RGB image. The preprocessing includes resizing the image to 224x224 pixels and normalizing the pixel values.
2. **Convolutional Layers:**
    - **Block 1:**
        - Two convolutional layers, each with 64 filters of size 3x3 and stride 1, padding 1.
        - Followed by a max-pooling layer with a 2x2 window and stride 2.
    - **Block 2:**
        - Two convolutional layers, each with 128 filters of size 3x3 and stride 1, padding 1.
        - Followed by a max-pooling layer with a 2x2 window and stride 2.

- **Block 3:**
  - Three convolutional layers, each with 256 filters of size 3x3 and stride 1, padding 1.
  - Followed by a max-pooling layer with a 2x2 window and stride 2.
- **Block 4:**
  - Three convolutional layers, each with 512 filters of size 3x3 and stride 1, padding 1.
  - Followed by a max-pooling layer with a 2x2 window and stride 2.
- **Block 5:**
  - Three convolutional layers, each with 512 filters of size 3x3 and stride 1, padding 1.
  - Followed by a max-pooling layer with a 2x2 window and stride 2.

3. **Fully Connected Layers:**
   - After the convolutional and pooling layers, the feature maps are flattened into a single vector.
   - Three fully connected layers are used:
     - The first two fully connected layers have 4096 neurons each.
     - The third fully connected layer has 2 neurons (for the 2 classes in the X-ray image dataset).
4. **Activation Functions:**
   - All convolutional and fully connected layers use the ReLU activation function.
   - The output layer uses the softmax activation function to produce probability distributions over the 2 classes.

2.8 Dataset

Chest X-Ray Image Dataset

The Chest X-ray Images (Pneumonia) dataset is a widely used collection of X-ray images that serves as a valuable resource for training and evaluating machine learning models aimed at detecting pneumonia. Here's a detailed review of the dataset:

1. **Overview**
   (a) **Source:** The dataset is available on mendeley data and was originally curated by the Guangzhou Women and Children's Medical Center.
   (b) **Purpose:** It is designed to assist in the development of machine learning models for the automatic detection of pneumonia from chest X-ray images.
2. **Dataset Composition**
   (a) **Total Images:** The dataset contains 5,863 chest X-ray images.
      i. **Normal Images:** 1,583 images classified as normal (healthy).
      ii. **Pneumonia Images:** 4,275 images classified as pneumonia.
3. **Data Splits**
   (a) **Training Set:**
      i. Contains 5,216 images, used to train the models.
      ii. Includes a mix of normal and pneumonia images.
   (b) **Validation Set:**
      i. Typically derived from the training set, it is used to tune model hyperparameters and validate performance.
   (c) **Testing Set:**
      i. Contains 624 images, used to evaluate the final model performance.
      ii. Separate from the training and validation sets to provide an unbiased performance metric.
4. **Image Characteristics**
   (a) **Resolution:** Images vary in resolution but are typically resized to a consistent dimension (224x224 pixel) during preprocessing for model training.
   (b) **Grayscale:** All images are in grayscale, representing different shades of gray to indicate various densities of tissue, bone, and fluids.
5. **Annotations and Labels**
   (a) **Annotations:** Each image is labeled as either "Normal" or "Pneumonia" based on clinical diagnosis.
   (b) **Label Quality:** The labels are considered reliable, as they are based on expert annotations from medical professionals.

6. **Preprocessing Steps**
    (a) **Normalization:** Pixel values are normalized to a standard range.
    (b) **Augmentation:** Data augmentation techniques such as rotation, flipping, and zooming are applied to increase the variability of the training data and prevent overfitting.
    (c) **Resizing:** Images are resized to 224X224 pixel fit the input requirements of CNN architectures.

Mendeley Data X-Ray Images (Pneumonia) dataset is a reliable and useful resource for constructing and testing machine learning models for pneumonia identification. Its well-annotated photos, together with its accessibility, make it a valuable resource for our major project in medical imaging and machine learning.

Test X-Ray Images Dataset

This dataset contains 41 chest X-ray pictures obtained from **Amicare Hospital** and organised into two folders: "Normal" and "Pneumonia". These photos provide an extra real-world validation set for assessing the effectiveness of machine learning models trained on larger datasets, such as the Mendeley Data Chest X-Ray images (Pneumonia) dataset. Here's a detailed review of the dataset:

1. **Dataset Composition**
    (a) **Total Images:** 41 chest X-ray images.
        i. **Normal Images:** 21 images classified as normal (healthy).
        ii. **Pneumonia Images:** 20 images classified as pneumonia.
2. **Image Characteristics**
    (a) **Resolution:** The resolution of the images varies, as they are directly obtained from hospital equipment. Standard preprocessing such as resizing (to 224x224) is necessary.
    (b) **Format:** The images are in JPEG format.
    (c) **Grayscale:** Similar to standard medical imaging practices, these images are in grayscale, showcasing various shades of gray that represent different tissue densities.
3. **Annotations and Labels**
    (a) **Normal Folder:** Contains 21 X-ray images of patients with no signs of pneumonia. These images serve as control data to benchmark against pneumonia-affected X-rays.

(b) **Pneumonia Folder:** Contains 20 X-ray images of patients diagnosed with pneumonia, either bacterial or viral. These images highlight the presence of pneumonia-specific features such as lung opacities and consolidations.

4. **Image Analysis and Quality**

   (a) **Image Quality:** The quality of images may vary due to differences in patient positioning, exposure settings, and radiography equipment. Variations in image clarity and contrast are expected.

   (b) **Label Accuracy:** The images have been annotated and labeled by medical professionals at Amicare Hospital. The labels are considered reliable, as they are based on clinical diagnosis and radiologist evaluations.

The testing dataset obtained from Amicare Hospital supplements the training data, providing a subset of chest X-ray images collected from real-world clinical settings. Comprising a mix of normal and pneumonia cases, each image is labelled for classification, enabling rigorous evaluation of model performance in real-world scenarios. Through meticulous validation using metrics such as accuracy, precision, recall, and F1-score, the testing dataset facilitates comprehensive evaluation of classification performance and diagnostic accuracy, ultimately contributing to advancements in healthcare diagnostics and patient care.

Chapter 3

Literature Review

3.1 Some relevant works

The purpose of this literature review is to provide a comprehensive analysis of existing research, technologies, and systems related to X-ray image classification and CNN architecture.

Convolutional Neural Network Based Classification of Patients with Pneumonia using X-ray Lung Images

The paper "Convolutional Neural Network Based Classification of Patients with Pneumonia using X-ray Lung Images" by Hicham Moujahid, et. al. presents a study on using convolutional neural networks (CNNs) to classify pneumonia from X-ray lung images.
Key Points:

1. **Objective:** Develop an automated system for classifying pneumonia in patients using X-ray lung images through CNNs via transfer learning.
2. **Methodology:**
   (a) **Data:** The dataset used is "Labeled Optical Coherence Tomography (OCT) and Chest X-Ray images for Classification"
   (b) **Model:** Implemented transfer learning using models such as VGG16 ,VGG19, NasNetMobile, ResNet50v2, InceptionResNetV2.
   (c) **Evaluation:** Model's have accuracy as follows:

i. **VGG16:** 96.81%
ii. **VGG19:** 96.58%
iii. **NasNetMobile:** 83.37%
iv. **ResNet152V2:** 96.35%
v. **InceptionResNetV2:** 94.87%

Results:
The different CNN models trained via transfer learning demonstrated high accuracy in distinguishing between pneumonia and non-pneumonia cases. The model's performance indicated that it is a reliable tool for pneumonia detection.

Pneumonia Detection Using CNN-based Feature Extraction

The paper "Pneumonia Detection Using CNN-based Feature Extraction" by Dimpy Varshni, et al. explores the application of convolutional neural networks (CNNs) for detecting pneumonia from chest X-ray images.
Key Points:

1. **Objective:** Develop an automated and efficient method to detect pneumonia using CNN-based feature extraction from chest X-ray images.
2. **Methodology:**
   (a) **Data:** The dataset used is ChestX-ray14 released by Wang et al. (2017).
   (b) **Model:** Implemented DenseNet-169 CNN model.
   (c) **Evaluation:** Model achieved AUC of 0.8002.

Results:
The proposed CNN-based model achieved high AUC in detecting pneumonia from chest X-ray images, which highlights model's effectiveness.

Automatic Detection of COVID-19 Infection from Chest X-ray Using Deep Learning

The paper "Automatic Detection of COVID-19 Infection from Chest X-ray using Deep Learning" by Kishore Medhia, Md. Jamil, and Md. Iftekhar Hussain focuses

on using deep learning, specifically convolutional neural networks (CNNs), to detect COVID-19 infections from chest X-ray images.

Key Points:

1. **Objective:** Develop an automatic, reliable method to diagnose COVID-19 using chest X-rays.
2. **Methodology:**
   (a) **Data:** Utilized publicly available chest X-ray image dataset from kaggle.
   (b) **Model:** Implemented a custom CNN model.
   (c) **Evaluation:** Model is able to detect covid-19 with 93% accuracy.

Results:
The deep learning model demonstrated high accuracy in distinguishing COVID-19 positive cases from negative ones, showing its potential as an effective supplementary diagnostic tool.

Explainable DCNN based chest X-ray image analysis and classification for COVID-19 pneumonia detection

The paper "Explainable DCNN based chest X-ray image analysis and classification for COVID-19 pneumonia detection" by Jie Hou and Terry Gao presents a deep convolutional neural network (DCNN) approach for analyzing and classifying chest X-ray images to detect COVID-19 pneumonia, with a focus on model explainability.
Key Points:

1. **Objective:** Develop a DCNN model for detecting COVID-19 pneumonia from chest X-ray images while ensuring that the model's decisions are interpretable.
2. **Methodology:**
   (a) **Data:** The dataset used is Chest X-Ray Images (covid-19).
   (b) **Model:** Implemented a DCNN architecture tailored for feature extraction and classification.

(c) **Evaluation:** Model's average accuracy is above 96%.

Results:
The DCNN model achieved high accuracy in detecting COVID-19 pneumonia from chest X-ray images. The inclusion of explainability techniques provided insights into the model's decision-making process, making it possible to understand and trust the predictions made by the model.

Chest X-ray image analysis and classification for COVID-19 pneumonia detection using Deep CNN

The paper "Chest X-ray image analysis and classification for COVID-19 pneumonia detection using Deep CNN" by Terry Gao and Grace Wang focuses on employing deep convolutional neural networks (CNNs) for the detection of COVID-19 pneumonia from chest X-ray images.
Key Points:

1. **Objective:** Develop a custom deep CNN model to accurately classify chest X-ray images and detect COVID-19 pneumonia.
2. **Methodology:**
   (a) **Data:** Historical X-rays data collected at Middlemore Hospital and some open-to-public coronavirus infectors' chest X-ray images
   (b) **Model:** Implemented a custom CNN architecture for feature extraction and classification.
   (c) **Evaluation:** Model's accuracy is above 91%.

Results:
The deep CNN model achieved high accuracy in detecting COVID-19 pneumonia from chest X-ray images. The study concludes that deep CNNs are highly effective for the automatic detection of COVID-19 pneumonia from chest X-ray images.

3.2 Summary of Literature Review

By comparing different datasets, techniques, and findings, it highlights the diverse approaches researchers are taking to tackle the challenges in this field.

Author	Title	Dataset	Techniques/Methods	Accuracy
Hicham Moujahid, et. al. (2020) [1]	Convolutional Neural Network Based Clas- sification of Patients with Pneumonia using X-ray Lung Images	Labeled Optical Coherence Tomography (OCT) and Chest X-Ray images for Classification	VGG16 ,VGG19, NasNetMobile, ResNet50v2, InceptionResNetV2.	The proposed model achieved following accuracy: VGG16: 96.81% VGG19: 96.58% NasNetMobile: 83.37% ResNet152V2: 96.35% InceptionResNetV2: 94.87%
Dimpy Varshni, et al. (2019) [2]	Pneumonia Detection Using CNN-based Feature Extraction	Chest X-Ray Images (covid-19)	DenseNet-169 CNN model	Model achieved AUC of 0.8002.
Md. Iftekhar Hussain et al. (2020) [3]	Automatic Detection of COVID-19 Infection from Chest X-ray Using Deep Learning	Chest X-ray images	Custom CNN model	Model is able to detect covid-19 with 93% accuracy.
Jie Hou and Terry Gao (2021) [4]	Explainable DCNN based chest X-ray image analysis and classification for COVID-19 pneumonia detection	Chest X-Ray Images (covid-19)	Implemented a DCNN architecture tailored for feature extraction and classification.	Model's average accuracy is above 96% for classification.
Terry Gao and Grace Wang (2020) [5]	Chest X-ray image analysis and classification for COVID-19 pneumonia detection using Deep CNN	Historical X-rays data collected at Middlemore Hospital and some open-to-public coronavirus infectors' chest X-ray images.	Custom CNN architecture for feature extraction and classification.	Model's overall accuracy for classification is above 91%.

Table 3.1: Summary of Literature Review on X-Ray Image Classification Systems

Chapter 4

Design and Implementation

4.1 Model Architecture

The final model leverages EfficientNetB7 via transfer learning for pneumonia detection from chest X-ray images, featuring a robust architecture designed for optimal performance.

4.1.1 Base Model Configuration

The configuration of the base model is as follows:

1. **EfficientNetB7:** The EfficientNetB7 model is selected as the base model due to its superior performance and efficiency in image classification tasks. It is pre-trained on the ImageNet dataset, providing a rich set of learned features.

2. **Include Top Layer:** The top classification layer of the EfficientNetB7 is excluded (include_top=False) to allow customization for the specific pneumonia detection task.

3. **Input Shape:** The model is configured to accept input images of size 224x224 pixels with 3 color channels (RGB), which aligns with the standard input size for EfficientNetB7.

4.1.2 Additional Layers and Pooling

Additional layers and pooling layers for the model are as follows:

1. **Global Max Pooling:** A Global Max Pooling layer is added after the base model to reduce the spatial dimensions of the feature maps. This layer

condenses the feature maps into a single vector by taking the maximum value from each feature map, thus reducing computational load and preventing overfitting.

2. **Batch Normalization:** A BatchNormalization layer follows the pooling layer. This layer standardizes the output of the base model, normalizing the activations to have a mean of zero and a standard deviation of one. This helps in stabilizing and accelerating the training process by mitigating internal covariate shift.

3. **Classification Layer:** Finally, a Dense layer with a softmax activation function is added. This layer maps the feature vectors to the desired number of classes (e.g., 'normal' and 'pneumonia'). The softmax activation function provides a probability distribution over the classes, making it suitable for multi-class classification.

4.1.3 Model Compilation

The model compilation is as follows:

1. **Optimizer:** The Adamax optimizer is chosen for its adaptive learning rate capabilities, which helps in efficiently navigating the parameter space and achieving convergence.

2. **Learning Rate:** The initial learning rate is set to 0.0001, ensuring a gradual and stable training process without disrupting the pre-trained weights significantly.

3. **Loss Function:** Categorical Crossentropy is used as the loss function, which is ideal for multi-class classification problems as it measures the divergence between the predicted probability distribution and the true distribution.

4. **Metrics:** The model is evaluated using multiple metrics, including accuracy, precision, and recall. These metrics provide a comprehensive assessment of the model's performance across different aspects of classification.

4.1.4 Fine-Tuning Strategy

The fine-tuning strategy for the model is as follows:

1. **Unfreezing Layers:** Initially, only the top layers of the EfficientNetB7 base model are trained. After a few epochs, deeper layers of the base model are unfrozen gradually.

2. **Fine-Tuning Threshold:** Layers up to a specified depth (finetune_at= 100) are kept frozen initially to retain the learned features from the pre-trained model while fine-tuning the top layers. After this threshold, the layers are unfrozen to further fine-tune the model.

4.1.5 Training and Validation Data

The dataset and data pipeline for the model is as follows:

1. **Dataset:** The model is trained on a dataset of chest X-ray images, which is augmented to improve generalization. The dataset contains images labelled as 'normal' or 'pneumonia'.

2. **Data Generators:** Training and validation data generators are used to load and preprocess the images in batches, applying transformations such as scaling and normalization.

4.1.6 Evaluation and Testing

Validation and testing for model performance are as follows:

1. **Validation:** The model's performance is validated on a separate set of images during training to monitor generalization and prevent overfitting.

2. **Testing:** After training, the model is tested on a hold-out test set to evaluate its final performance. Performance metrics like accuracy, precision, recall, and F1-score are calculated.

This detailed design outlines the steps and considerations for building, training, and deploying a deep learning model based on EfficientNetB7 for pneumonia detection from chest X-ray images. It includes the architectural design, training strategy, and evaluation metrics. Same architecture is used for VGG16 used as a benchmark model, ResNet50v2, and EfficientNetb3.

4.2 Hardware and Software Requirements

4.2.1 Hardware Requirements

Hardware requirements for the project are as follows:

1. **Processor:** Intel Core i9-13th gen.

2. **Memory (RAM):** 16 GB or above.
3. **Graphics Processing Unit (GPU):** Nvidia Geforce RTX 4070.
4. **Storage:** 30 GB or above.

4.2.2 Software Requirements

Software requirements for the project are as follows:

1. **Operating System:** Windows 11 (64-bit).
2. **Deep Learning Framework:** TensorFlow, Sklearn.
3. **Server Framework:** Flask.
4. **Android Development:** Android Studio.
5. **Programming Language:** Python 3.8, Java.
6. **Libraries and Dependencies:** os, librosa, numpy, pandas, seaborn, matplotlib, tesorflow, sklearn, flask.

4.3 Implementation Details

Mainly three stages make up the implementation process:

1. **Phase 1:** Training phase
2. **Phase 2:** Testing phase
3. **Phase 3:** Deployment phase

4.3.1 Training Phase

The major steps associated with the training phases are as follows:

1. Configure Local workspace and Jupyter Notebook.
2. Install required Python libraries and frameworks in the conda environment.
3. Load **Chest X-ray Images** dataset on with test, train, validate folder.
4. Exploratory data analysis for **Chest X-ray Images** dataset.

5. Images are pre-processed to a size of 224X224 pixel with 3 Channel, Then image are augmented by horizontal, vertical and rotational flips.
6. Splitting data in a ratio of 90:10 in training and testing parts.
7. Sequential transfer learning enabled network model is defined.
8. Model is trained on training data for 50 epochs and with a batch size of 32.
9. Subsequent trained model is saved as "Final_EfficientNetB7_new_model.h5" in the current working directory and weights are saved in the same working directory as Final_EfficientNetB7_new_weights.h5.
10. Trained model is then evaluated on the testing data.
11. Model Accuracy parameters are then calculated for the trained model and evaluated and subsequently the model is fine-tuned.

The same process is followed to train the VGG16, EfficientNetB3, ResNet50v2 based models.

4.3.2 Testing Phase

The major steps associated with the testing phases are as follows:

1. Configure Local workspace and Jupyter Notebook.
2. Load the pre-trained EfficientNetB7 model and its weights for prediction.
3. Loading Test dataset(Normal/Pneumonia) collected from amicare hospital.
    (a) Load and preprocess the image (resize to 224x224 pixels and convert to array).
    (b) Make a prediction using the loaded model.
    (c) Extract the predicted label and the confidence level.
    (d) Assume true labels are 'Pneumonia' (label 1) for all images in this directory.
    (e) Print predicted category and confidence level.
    (f) Append the filename, predicted label, and confidence to an Excel sheet.
4. Calculate the confusion matrix, accuracy, precision, recall, and F1 score using true and predicted labels and print these metrics to the console for review.

5. Append the calculated metrics to a new sheet in the Excel workbook for documentation and further analysis.

6. Print Confusion Matrix and metrics.

4.3.3 Implementation Phase

The Android application offers an easy user experience for healthcare professionals to upload chest X-ray images and receive diagnostic predictions for pneumonia. The application communicates with a Flask backend server that has trained deep-learning models for image classification. This architecture ensures safe user authentication and reliable image analysis by using the models such as EfficientNetB7, VGG16, ResNet50v2, and EfficientNetB3.

Implementation Design for the Android Application with Flask Backend

1. **User Interface (UI) Design**
   (a) **Login Screen**
      i. **Fields:**
         A. **Username:** Text input for the user's username.
         B. **Password:** Password input for the user's password.
      ii. **Buttons:**
         A. **Login:** Triggers the login process.
         B. **Sign Up:** Navigates to the Sign-Up screen.
      iii. **Functionality:** Allows users to authenticate themselves by sending login credentials to the Flask server. Displays error messages for incorrect credentials or missing fields.
   (b) **Sign-Up Screen**
      i. **Fields:**
         A. **Username:** Text input for the new user's desired username.
         B. **Password:** Password input for the user's password.
      ii. **Buttons:**
         A. **Sign Up:** Registers the user.
         B. **Back to Login:** Navigates back to the Login screen.
      iii. **Functionality:** Allows new users to create an account and checks for existing usernames if a user exists raise a toast to change the username for sign-up.

(c) **Home Screen**

i. **Components:**

A. **Redirect Buttons:** Navigates to subsequent upload screen for the selected model for image classification

ii. **Functionality:** Main interface where users can navigate to the different model screens.

(d) **Upload and Result Screen**

i. **Components:**

A. **Launch Gallery:** Allows users to select an image from their device.

B. **Launch Camera:** Allows users to click an image from their device.

C. **Image Preview:** Displays the selected image before uploading.

D. **Prediction Result:** Shows the classification result (e.g., Normal, Pneumonia).

E. **Confidence Score:** Displays the confidence level of the prediction.

ii. **Functionality:** Ensures the selected file is a valid image format (e.g., JPG, PNG) and uploads it to the Flask server for analysis and provides feedback to the user about the diagnostic prediction made by the model.

2. **Networking**

(a) **HTTP Client:** OkHttp is used for making HTTP requests to the Flask backend.

(b) **POST Requests:** Used for sending login and signup data as JSON.

(c) **Multipart Form Data:** Used for sending image files during upload.

3. **Image Upload and Prediction**

(a) **Image Preprocessing**

i. **Bitmap and ImageUtils:** Resize and format images to the required input size (224x224 pixels) before uploading.

(b) **OkHttp Multipart Requests**

i. **Multipart Requests:** Sends images as multipart/form data to the Flask server.

ii. **File Validation:** Ensures that only valid image files are uploaded.

(c) **Prediction Handling**

i. **Flask Endpoints:** Different endpoints for each model (EfficientNetB7, VGG16, ResNet50v2, EfficientNetB3).

ii. **Response Parsing:** Parses the prediction result and confidence score returned from the server and prints them in respective textviews.

4. **Flask Backend Integration**

(a) **API Endpoints**

i. **/login:** Validates user credentials and returns a session token.

ii. **/signup:** Registers new users and stores hashed passwords in the database.

iii. **/logout:** Ends the user session and invalidates the session token.

iv. **/efficientnetb7, /vgg16, /resnet50v2, /efficientnetb3:** Handles image uploads and returns predictions.

(b) **Model Loading**

i. **Keras Models:** Loads trained models (EfficientNetB7, VGG16, ResNet50v2, EfficientNetB3) with their respective weights.

ii. **Model Prediction:** Processes the uploaded image and generates predictions.

(c) **File Handling**

i. **Secure Filename Handling:** Ensures uploaded files are saved with secure filenames.

ii. **Uploads Folder:** Stores uploaded images for processing and future reference.

5. **Database: SQLite**

(a) **User Model:** Stores user information, including username and hashed password.

(b) **User Table:** Columns for ID, username, and password.

(c) **SQLAlchemy:** ORM for handling database operations in Flask.

6. **Deployment**

(a) **Ngrok**

i. **Tunnel Configuration:** Exposes the local Flask server to the internet using a public URL.

ii. **Ngrok Integration:** Ngrok service running on the host for the server, redirects all requests to the flask server.

(b) **Flask Server**

i. **Production Server:** Server hosted communicates with the android application through ngrok tunnel.

ii. **Environment Configuration:** Python environment required is managed through conda which manages all libraries required for the server to run.

This design outlines the architecture and components required to create an Android app with a Flask backend for identifying chest X-ray images. It addresses UI design, networking, image upload, backend integration, database, and deployment. This meticulous design offers a secure, user-friendly, and efficient application to assist healthcare practitioners in diagnosing pneumonia based on X-ray images.

4.4 Application Overview

4.4.1 Backend Server

Flask server and ngrok tunnel in operation are shown below:

```
 * Running on http://national-pleasantly-earwig.ngrok-free.app
 * Traffic stats available on http://127.0.0.1:4040
127.0.0.1 - - [09/Jun/2024 17:06:40] "GET /serverstat HTTP/1.1" 200 -
127.0.0.1 - - [09/Jun/2024 17:06:42] "POST /login HTTP/1.1" 200 -
127.0.0.1 - - [09/Jun/2024 17:06:49] "POST /login HTTP/1.1" 200 -
1/1 [==============================] - 0s 43ms/step
[[9.9999988e-01 1.0436665e-07]]
127.0.0.1 - - [09/Jun/2024 17:06:55] "POST /efficientnetb7 HTTP/1.1" 200 -
1/1 [==============================] - 0s 23ms/step
[[9.9952447e-01 4.7548726e-04]]
127.0.0.1 - - [09/Jun/2024 17:07:02] "POST /efficientnetb3 HTTP/1.1" 200 -
127.0.0.1 - - [09/Jun/2024 17:07:07] "GET /logout HTTP/1.1" 200 -
```

Figure 4.1: Flask Server

```
Session Status                online
Account                       uddhav gupta (Plan: Free)
Update                        update available (version 3.10.1, Ctrl-U to update)
Version                       3.9.0
Region                        India (in)
Latency                       73ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://national-pleasantly-earwig.ngrok-free.app -> http://localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              6       0       0.06    0.02    0.05    1.21

HTTP Requests
-------------

GET  /logout                  200 OK
POST /efficientnetb3          200 OK
POST /efficientnetb7          200 OK
POST /login                   200 OK
POST /login                   200 OK
GET  /serverstat              200 OK
```

Figure 4.2: Ngrok Tunnel

4.4.2 Android Application

Android Application interface in the production environment is as follows:

Sign Up activity for user registration:

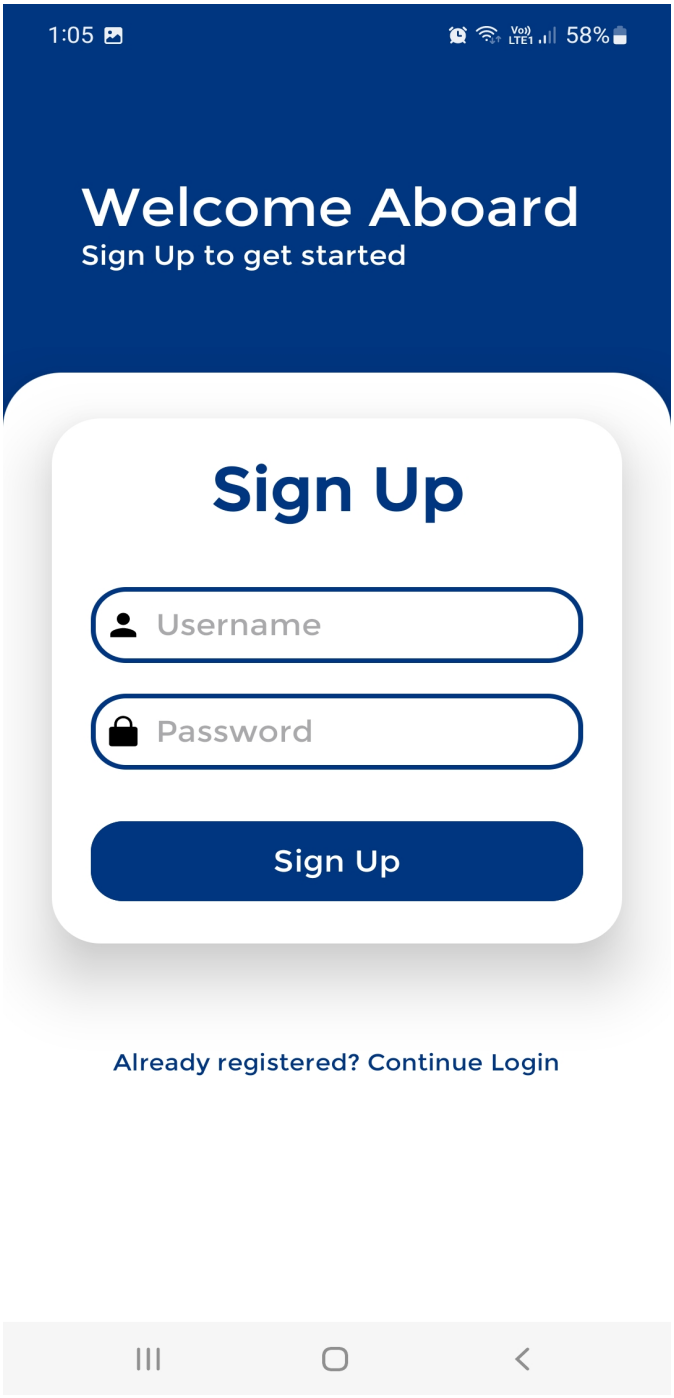

Figure 4.3: Sign Up Activity

Login activity for user login:

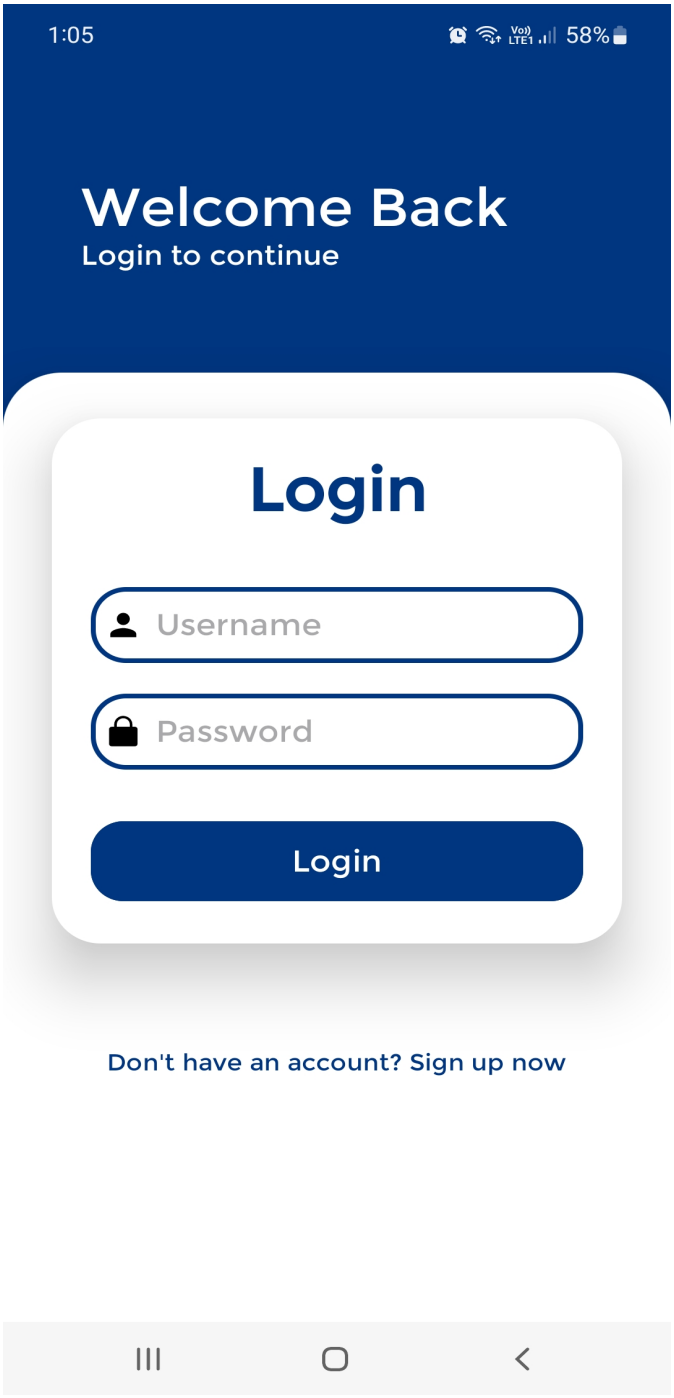

Figure 4.4: Login Activity

Home Fragment for application home page:

Figure 4.5: Home Fragment

Navigation Drawer for application navigation:

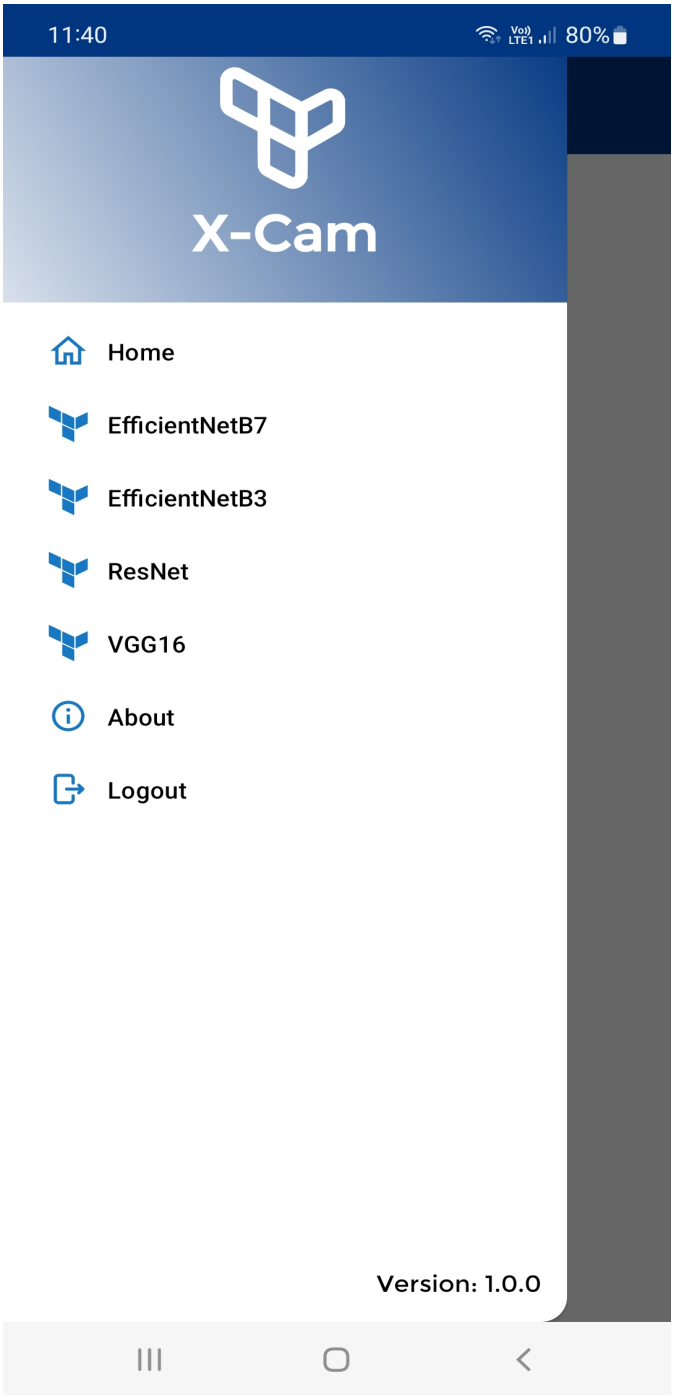

Figure 4.6: Navigation Drawer

EfficientNetB7 activity for X-ray image prediction:

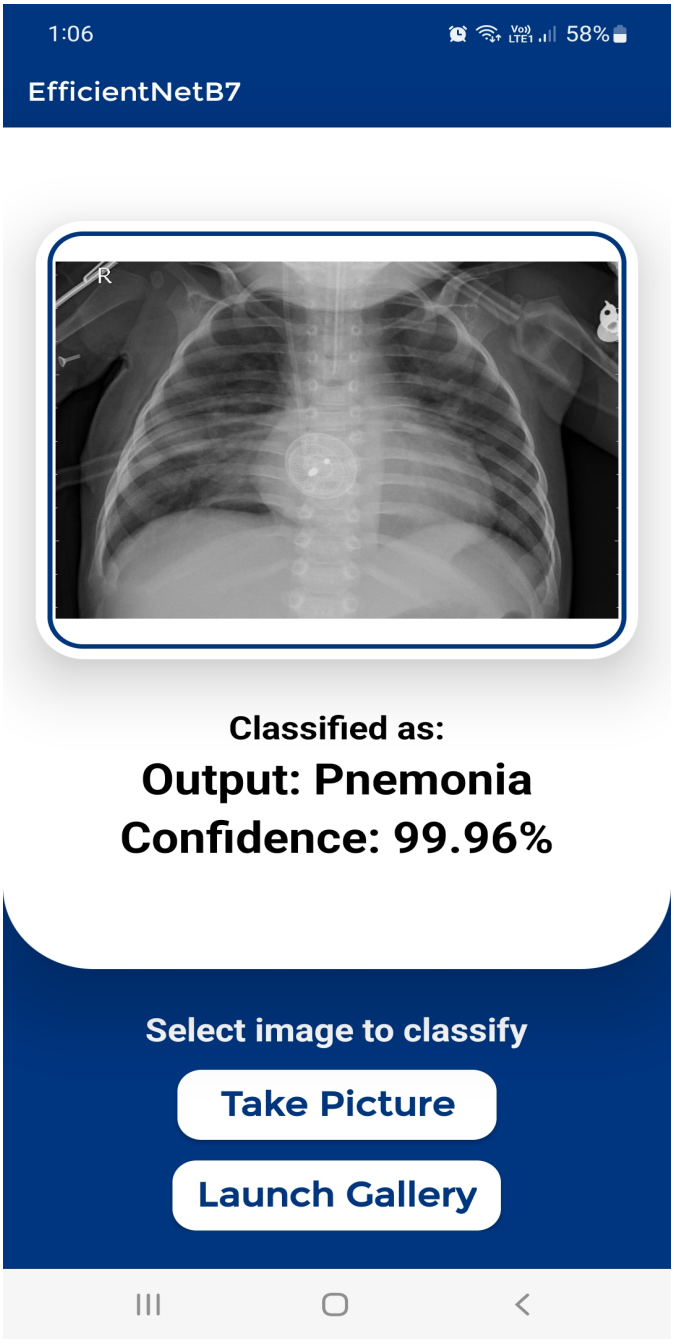

Figure 4.7: EfficientNetb7 Activity

EfficientNetB3 activity for X-ray image prediction:

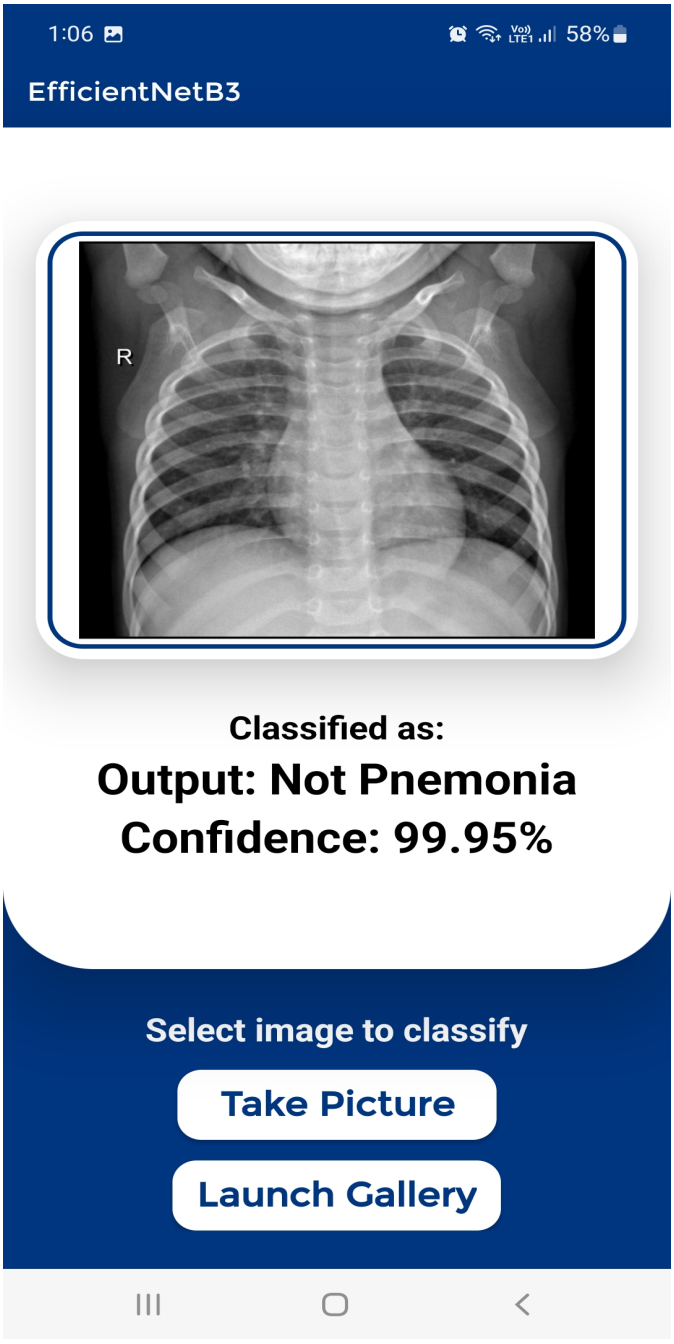

Figure 4.8: EfficientNetb3 Activity

VGG16 activity for X-ray image prediction:

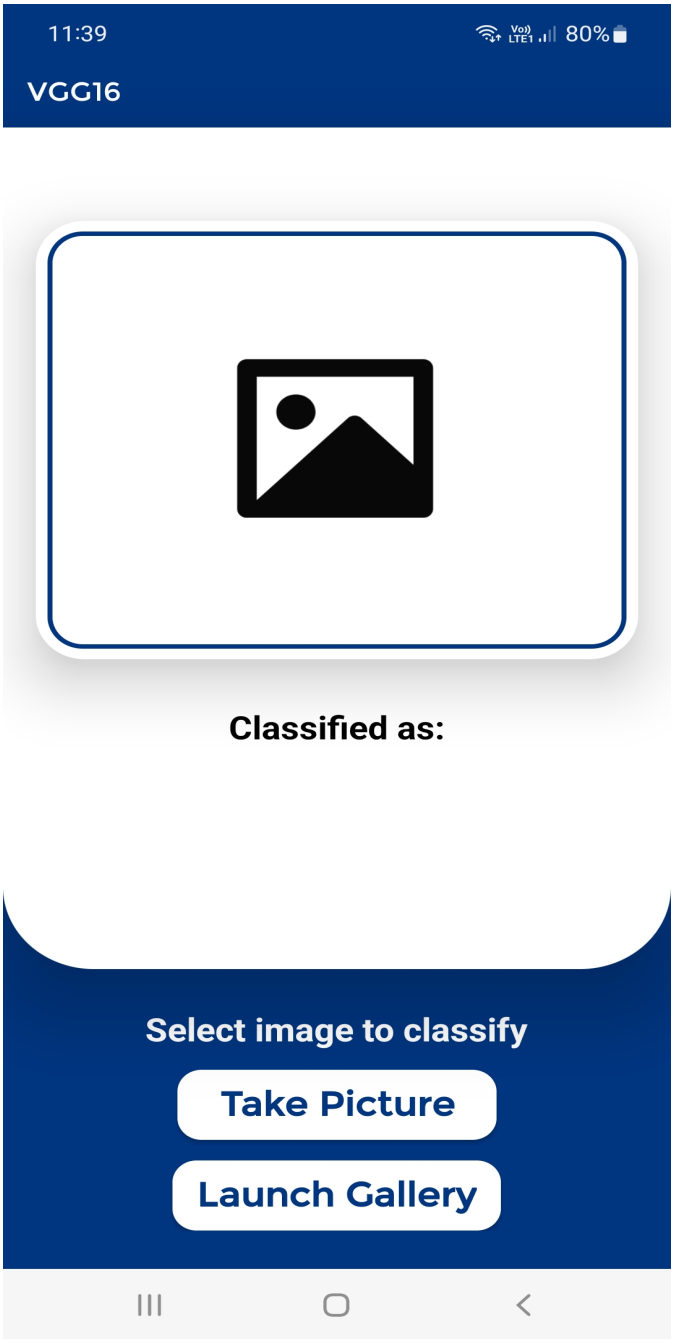

Figure 4.9: VGG16 Activity

ResNet50v2 activity for X-ray image prediction:

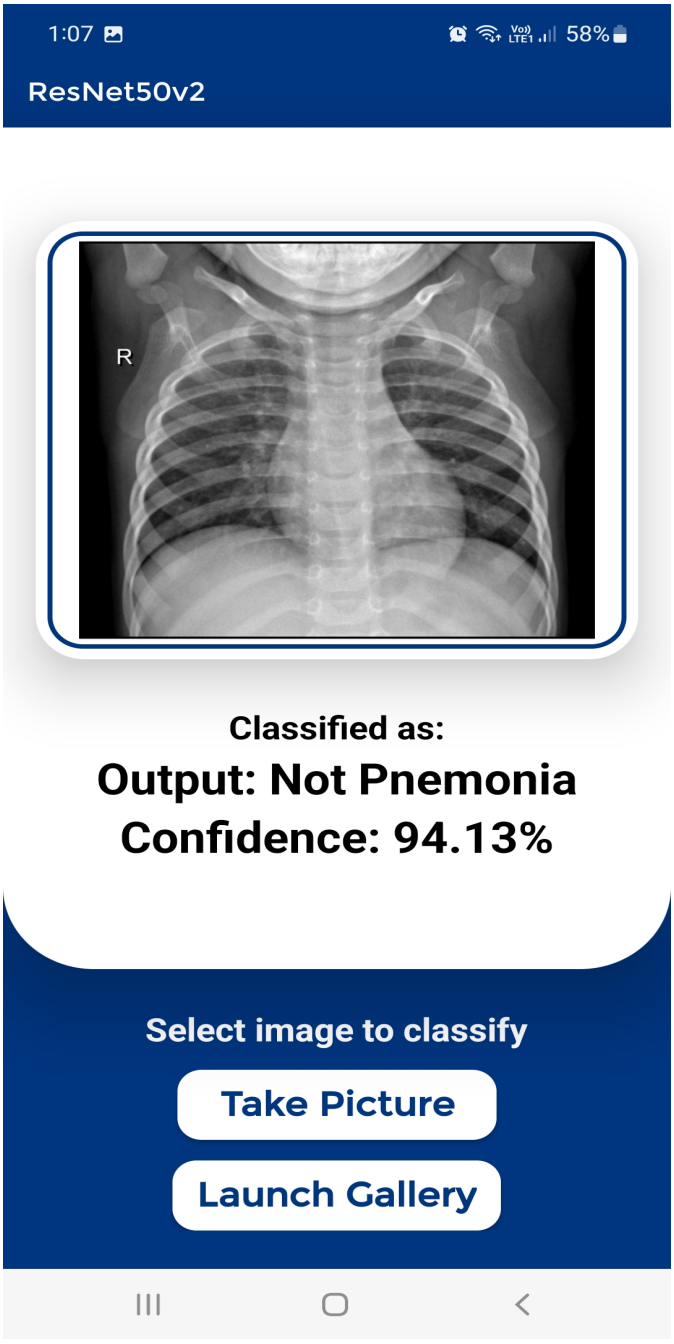

Figure 4.10: ResNet50v2 Activity

About activity:

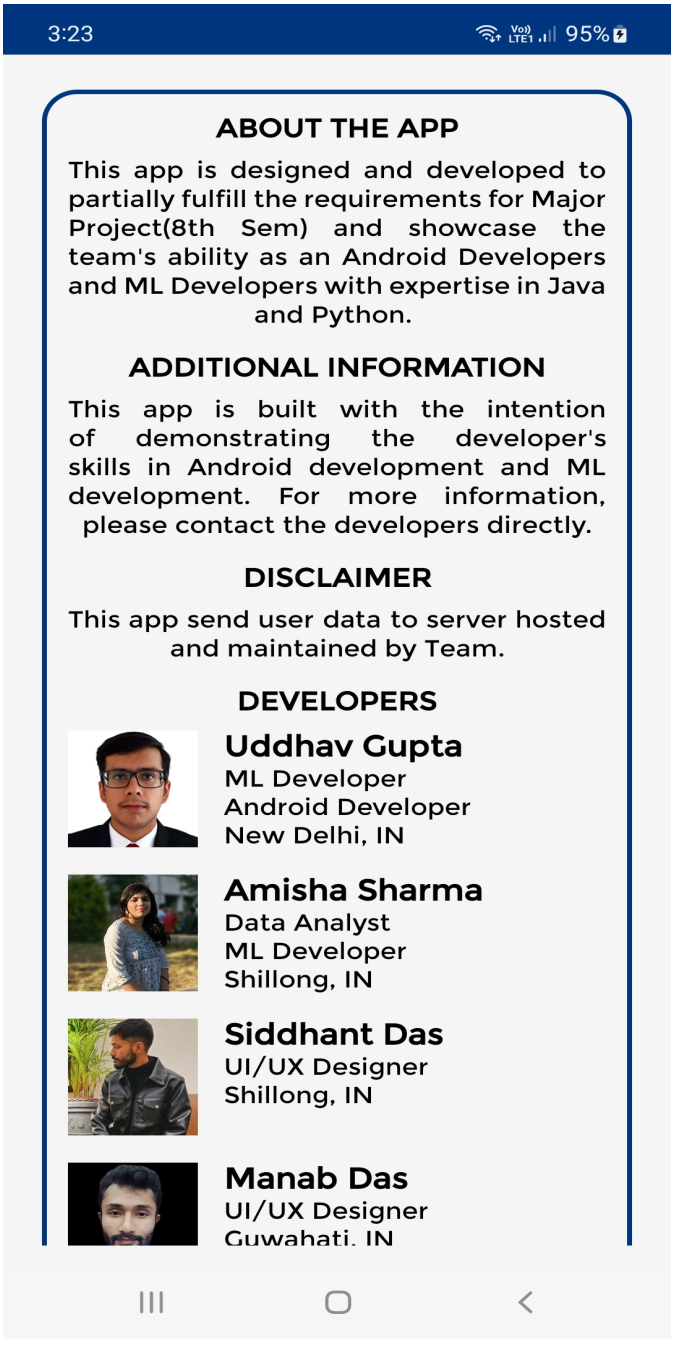

Figure 4.11: About Activity

Chapter 5

Result and Analysis

5.1 Performance Evaluation

The table provides a comparative analysis of four different models: EfficientNetB7, EfficientNetB3, VGG16, and ResNet50v2, based on their performance in classifying pneumonia from chest X-ray images. The performance metrics include training accuracy, test accuracy, validation accuracy, precision, recall, and F1-score for both the 'Normal' and 'Pneumonia' categories.

Outcomes from the proposed models for X-ray image classification are as follows:

	EfficientNetB7	**EfficientNetB3**	**VGG16**	**ResNet50v2**
Train Accuracy	**1.0000**	0.9806	0.9773	0.9284
Test Accuracy	**0.9830**	0.9830	0.9659	0.8920
Val Accuracy	**0.9904**	0.9713	0.9761	0.8971
Training Loss	**0.0011**	0.0146	0.0856	0.3831
Precision	-	-	-	-
Normal	0.92	0.90	0.96	**1.00**
Pneumonia	**0.99**	0.98	0.97	0.90
Recall	-	-	-	-
Normal	**0.97**	0.95	0.90	0.65
Pneumonia	0.97	0.97	0.99	**1.00**
F1-Score	-	-	-	-
Normal	**0.95**	0.93	0.93	0.79
Pneumonia	**0.98**	0.98	0.98	0.95

Table 5.1: Performance comparison of different models for pneumonia detection

Detailed results for training of models are as follows:

VGG16 (Benchmark)

Evaluation for VGG16 are as follows:

Analysis: VGG16 demonstrates high performance with strong accuracies across training, test, and validation datasets. The model shows high precision, recall, and F1-scores, particularly excelling in pneumonia detection with a recall of 0.99 and an F1-score of 0.98.

Metric	**Value**
Training Accuracy	97.73%
Test Accuracy	96.59%
Validation Accuracy	97.61%
Precision	
Normal	0.96
Pneumonia	0.97
Recall	
Normal	0.90
Pneumonia	0.99
F1-Score	
Normal	0.93
Pneumonia	0.98

Table 5.2: Evaluation Metrics for VGG16 (Benchmark)

The loss consistently decreases with occasional spikes, indicating that the model is learning well from the training data. With same trends followed for validation loss. The accuracy steadily increases and stabilizes around 96-97% towards the end, indicating that the model is performing well on the training data.The validation accuracy shows a similar upward trend with fluctuations, also stabilizing around 96-97%.
The graphs show the training and validation loss as well as the training and validation accuracy of a deep learning model over multiple epochs:

Figure 5.1: Loss and Accuracy for VGG16

EfficientNetB7

Evaluation for EfficientNetB7 are as follows:

Comparison with VGG16: EfficientNetB7 surpasses VGG16 in all accuracy metrics, achieving perfect training accuracy and higher test and validation accuracies. While the precision for normal cases is slightly lower than VGG16, the recall and F1-scores are comparable, maintaining high performance for both categories.

Metric	**Value**
Training Accuracy	100%
Test Accuracy	98.30%
Validation Accuracy	99.04%
Precision	
Normal	0.92
Pneumonia	0.99
Recall	
Normal	0.97
Pneumonia	0.97
F1-Score	
Normal	0.95
Pneumonia	0.98

Table 5.3: Evaluation Metrics for EfficientNetB7

The loss consistently decreases with occasional spikes, indicating that the model is learning well from the training data. With same trends followed for validation loss. The accuracy steadily increases and stabilizes around 98-100% towards the end, indicating that the model is performing well on the training data.The validation accuracy shows a similar upward trend with fluctuations, also stabilizing around 98-100%.
The graphs show the training and validation loss as well as the training and validation accuracy of a deep learning model over multiple epochs:

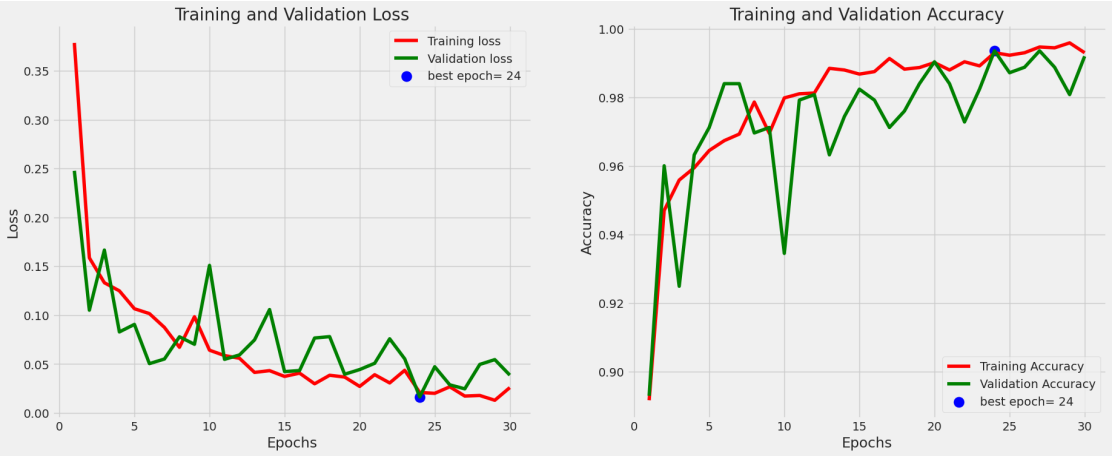

Figure 5.2: Loss and Accuracy for EfficientNetB7

EfficientNetB3

Evaluations for EfficientNetB7 are as follows:

Comparison with VGG16: EfficientNetB3 also shows higher training and test accuracies compared to VGG16. Its validation accuracy is slightly lower than VGG16 but still strong. The precision, recall, and F1-scores are comparable, with slight variations favoring EfficientNetB3 in pneumonia detection.

The loss consistently decreases with occasional spikes and a very minor increase in the end. With same trends followed for validation loss. The accuracy steadily increases and stabilizes around 97-98% towards the end, indicating that the model is performing well on the training data.The validation accuracy shows a similar upward trend with fluctuations, also stabilizing around 97-98%.

Metric	Value
Training Accuracy	98.06%
Test Accuracy	98.30%
Validation Accuracy	97.13%
Precision	
Normal	0.90
Pneumonia	0.98
Recall	
Normal	0.95
Pneumonia	0.97
F1-Score	
Normal	0.93
Pneumonia	0.98

Table 5.4: Evaluation Metrics for EfficientNetB3

The graphs show the training and validation loss as well as the training and validation accuracy of a deep learning model over multiple epochs:

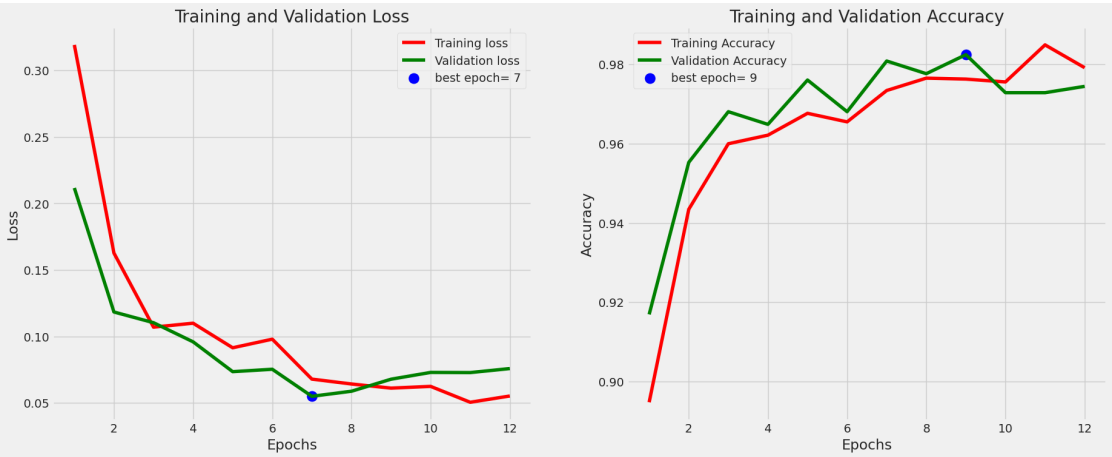

Figure 5.3: Loss and Accuracy for EfficientNetB3

ResNet50v2

Evaluations for ResNet50v2 are as follows:

Comparison with VGG16: ResNet50v2 shows lower performance across all metrics compared to VGG16. Despite having perfect precision for normal cases and perfect recall for pneumonia, its overall accuracy and F1-scores are notably

lower. This indicates a weaker ability to generalize and accurately classify both categories.

Metric	**Value**
Training Accuracy	92.84%
Test Accuracy	89.20%
Validation Accuracy	89.71%
Precision	
Normal	1.00
Pneumonia	0.90
Recall	
Normal	0.65
Pneumonia	1.00
F1-Score	
Normal	0.79
Pneumonia	0.95

Table 5.5: Evaluation Metrics for ResNet50v2

The loss consistently decreases with occasional spikes and a very minor increase in the end. With same trends followed for validation loss. The accuracy steadily increases and stabilizes around 89-92% towards the end, indicating that the model is performing well on the training data.The validation accuracy shows a similar upward trend with fluctuations, also stabilizing around 89-92%.
The graphs show the training and validation loss as well as the training and validation accuracy of a deep learning model over multiple epochs:

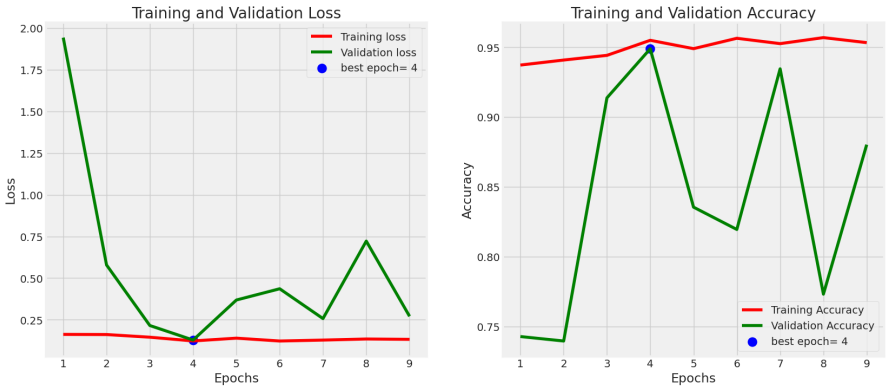

Figure 5.4: Loss and Accuracy for ResNet50v2

EfficientNetB7 stands out as the best model for pneumonia detection.

5.1.1 Confusion Matrix

The confusion matrix shows the performance of a machine learning model on test data for the x-ray image classification problem. The rows of the table represent the actual classes of the data points, and the columns represent the predicted classes. Each cell of the table contains the number of data points that were assigned to a particular predicted class, given their actual class.
The confusion matrix can be used to calculate some different metrics to evaluate the performance of the model, such as accuracy, precision, recall, and F1 score.
Accuracy is the proportion of all data points that were correctly classified by the model. It is calculated by summing the diagonal elements of the confusion matrix and dividing by the total number of data points.

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

Recall is the proportion of data points that are actually in a particular class that were predicted to be in that class. It is calculated by dividing the number of true positives for a particular class by the total number of data points that are actually in that class.

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

Precision is the proportion of data points that were predicted to be in a particular class and were actually in that class. It is calculated by dividing the number of true positives for a particular class by the total number of data points that were predicted to be in that class.

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

The **F1 score** is a harmonic mean of precision and recall.

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

EfficientNetb7 Confusion Matrix

Overall, the confusion matrix for EfficientNetb7 shows that the proposed model is performing well on this classification problem. The accuracy is notably high, and the precision, recall, and F1 scores are also high.

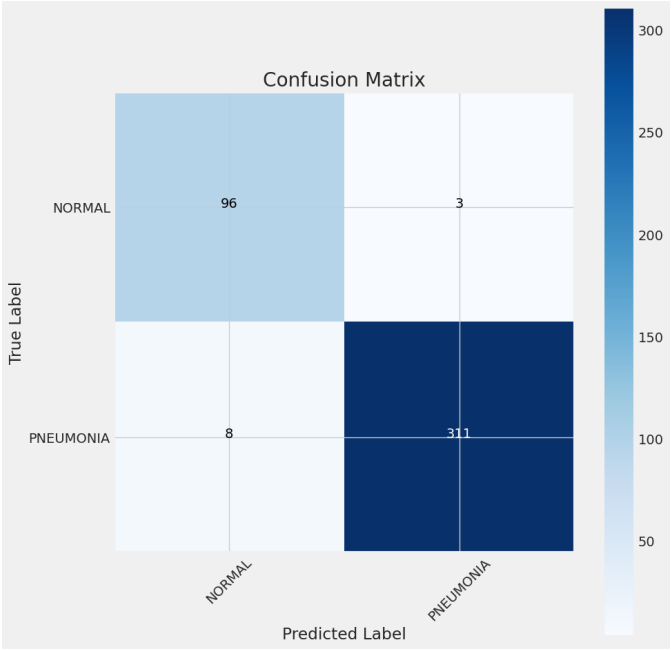

Figure 5.5: Confusion Matrix for EfficientNetB7

	precision	recall	f1-score	support
NORMAL	0.92	0.97	0.95	99
PNEUMONIA	0.99	0.97	0.98	319
accuracy			0.97	418
macro avg	0.96	0.97	0.96	418
weighted avg	0.97	0.97	0.97	418

Figure 5.6: Results for EfficientNetB7

EfficientNetb3 Confusion Matrix

Overall, the confusion matrix for EfficientNetb3 shows that the proposed model is performing well on this classification problem. The accuracy is high, and the precision, recall, and F1 scores are also high.

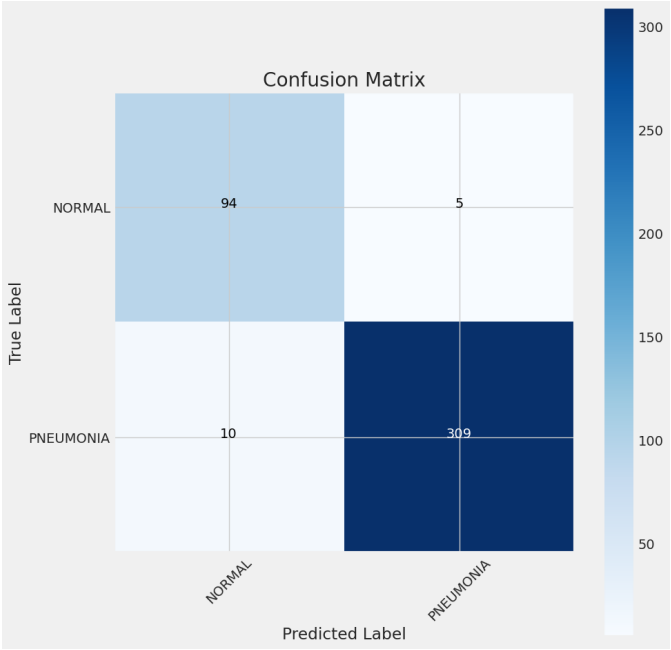

Figure 5.7: Confusion Matrix for EfficientNetB3

	precision	recall	f1-score	support
NORMAL	0.90	0.95	0.93	99
PNEUMONIA	0.98	0.97	0.98	319
accuracy			0.96	418
macro avg	0.94	0.96	0.95	418
weighted avg	0.97	0.96	0.96	418

Figure 5.8: Results for EfficientNetB3

VGG16 Confusion Matrix

Overall, the confusion matrix for VGG16 shows that the proposed model is performing well on this classification problem. The accuracy is high, and the precision, recall, and F1 scores are also high.

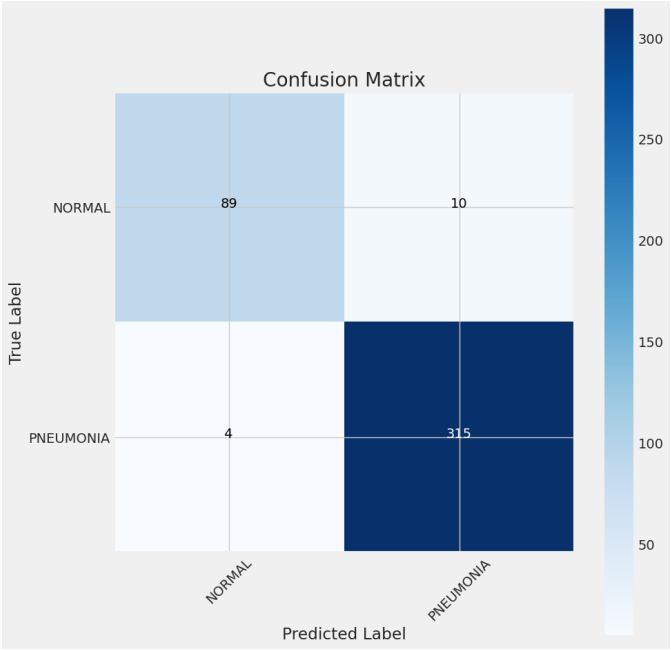

Figure 5.9: Confusion Matrix for VGG16

	precision	recall	f1-score	support
NORMAL	0.96	0.90	0.93	99
PNEUMONIA	0.97	0.99	0.98	319
accuracy			0.97	418
macro avg	0.96	0.94	0.95	418
weighted avg	0.97	0.97	0.97	418

Figure 5.10: Results for VGG16

ResNet50v2 Confusion Matrix

Overall, the confusion matrix for ResNet50v2 shows that the proposed model is slightly underperforming on this classification problem. The accuracy is satisfactory, and the precision, recall, and F1 scores are also moderate.

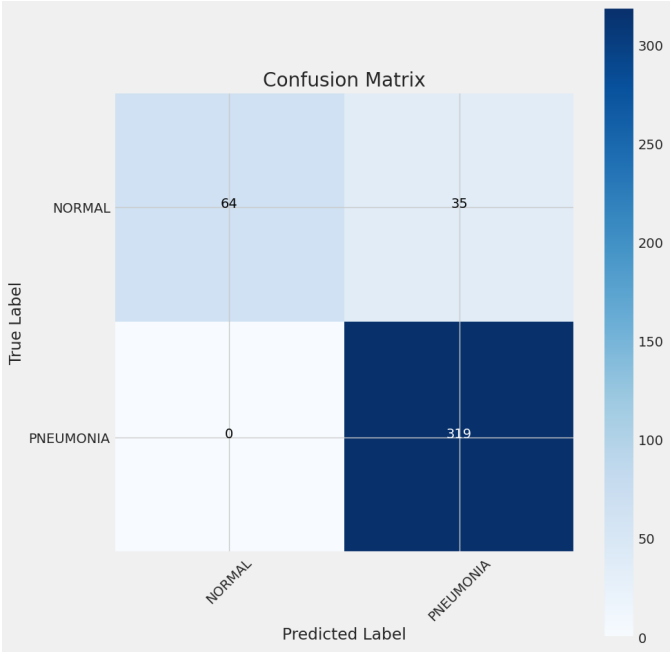

Figure 5.11: Confusion Matrix for ResNet50v2

	precision	recall	f1-score	support
NORMAL	1.00	0.65	0.79	99
PNEUMONIA	0.90	1.00	0.95	319
accuracy			0.92	418
macro avg	0.95	0.82	0.87	418
weighted avg	0.92	0.92	0.91	418

Figure 5.12: Results for ResNet50v2

5.2 Test Data Evaluation

The evaluation of the model's performance on the test data provides insights into its classification accuracy and confidence levels for both "Pneumonia" and "Normal" cases. The analysis is based on the provided classification results and confusion matrices.

5.2.1 Confusion Matrix Analysis

The confusion matrix for "Normal" test data and "Pneumonia" test data offers a comprehensive overview of the model's performance:

Normal Test Data (Not Pneumonia) Confusion Matrix:

Results for Normal class test data are as follows:

- Accuracy: 0.90
- Precision: 1.00
- Recall: 0.90
- F1 Score: 0.95

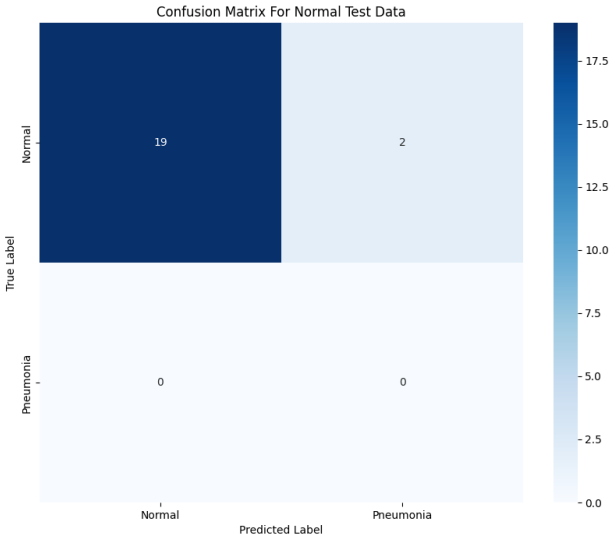

Figure 5.13: Confusion Matrix for Normal test data

This indicates the model correctly identified 19 "Normal" cases and incorrectly classified 2 "Normal" cases as "Pneumonia".

Pneumonia Test Data Confusion Matrix:

- Accuracy: 0.90
- Precision: 1.00
- Recall: 0.95
- F1 Score: 0.97

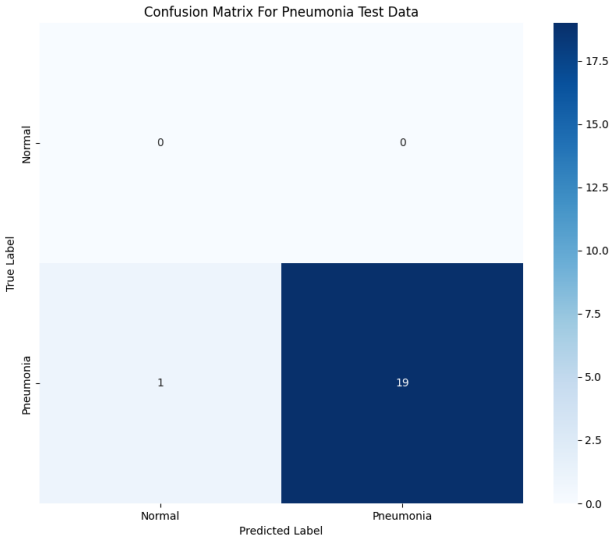

Figure 5.14: Confusion Matrix for Pneumonia test data

This indicates the model correctly identified 19 "Pneumonia" cases and incorrectly classified 1 "Pneumonia" cases as "Normal".

The overall model performance demonstrates high accuracy and confidence in classifying both "Pneumonia" and "Not Pneumonia" cases. Most images are classified correctly with high confidence. However, there are a few misclassifications, indicating areas where the model could be further refined to improve accuracy.

The confusion matrices indicate a high number of true positives for both classes, with a few false positives and false negatives. This suggests that while the model is generally effective, there is room for improvement, especially in reducing the

number of misclassifications and increasing the confidence scores for borderline cases.

Test Data Analysis

The classification results and confidence scores for individual images provide further detail on the model's performance.

Image-wise analysis for normal test cases with their respective confidences:

File Name	Classification	Confidence
l1.jpg	Not Pneumonia	100
l2.jpg	Not Pneumonia	98.88
l3.jpg	Not Pneumonia	99.22
l4.jpg	Not Pneumonia	82.79
l5.jpg	Not Pneumonia	97.33
l6.jpg	Pneumonia	98.21
l7.jpg	Pneumonia	88.73
l8.jpg	Not Pneumonia	99.94
l9.jpg	Not Pneumonia	89.42
l10.jpg	Not Pneumonia	98.98
l11.jpg	Not Pneumonia	51.86
l12.jpg	Not Pneumonia	87.67
l13.jpg	Not Pneumonia	96.51
l14.jpg	Not Pneumonia	100
l15.jpg	Not Pneumonia	52.98
l16.jpg	Not Pneumonia	99.67
l17.jpg	Not Pneumonia	73.2
l18.jpg	Not Pneumonia	96.78
l19.jpg	Not Pneumonia	98.56
l20.jpg	Not Pneumonia	100
l21.jpg	Not Pneumonia	87.25

Table 5.6: Classification and Confidence Scores for Normal Test Images

Out of 21 images, images l6 and l7 were wrongly classified as Pneumonia while they were actually normal images.

Image-wise analysis for pneumonia test cases with their respective confidences:

File Name	**Classification**	**Confidence**
p1.jpg	Pneumonia	56.99
p2.jpg	Pneumonia	98.02
p3.jpg	Pneumonia	100
p4.jpg	Pneumonia	99.72
p5.jpg	Pneumonia	100
p6.jpg	Pneumonia	60.04
p7.jpg	Pneumonia	99.63
p8.jpg	Pneumonia	99.7
p9.jpg	Pneumonia	98.17
p10.jpg	Pneumonia	100
p11.jpg	Pneumonia	99.97
p12.jpg	Pneumonia	99.9
p13.jpg	Pneumonia	99.96
p14.jpg	Pneumonia	100
p15.jpg	Pneumonia	99.97
p16.jpg	Pneumonia	100
p17.jpg	Pneumonia	97.95
p18.jpg	Pneumonia	99.52
p19.jpg	Pneumonia	100
p20.jpg	Not Pneumonia	98.59

Table 5.7: Classification and Confidence Scores for Pneumonia Test Images

Out of 20 images, image p20 was wrongly classified as a Normal image while it was a Pneumonia image.

Chapter 6

Conclusions

6.1 Conclusion

This project aimed to develop a robust and efficient deep learning model for pneumonia detection using chest X-ray images. By leveraging different convolutional neural network architectures, we addressed critical issues in medical image classification, such as accuracy, efficiency, and ease of deployment. Initially, we used the VGG16 architecture as a benchmark, known for its simplicity and effectiveness in feature extraction. However, its computational demands and limited performance on larger datasets necessitated exploration of more advanced models.

Our final model, EfficientNetB7, demonstrated superior performance, achieving higher accuracy and better generalization compared to VGG16, EfficientNetB3, and ResNet50V2. EfficientNetB7's innovative scaling methods—balancing network depth, width, and resolution—proved particularly effective for this task, offering a significant improvement in classification capabilities while maintaining manageable computational requirements.

The project also included the development of an Android application and a Flask server to facilitate the deployment and accessibility of the model. The Android app allows users to easily upload chest X-ray images and receive diagnostic predictions in real-time, making the system practical for use in various healthcare settings. The Flask server serves as the backend, handling image processing and model inference, ensuring smooth and efficient operation.

Overall, this project successfully addressed key challenges in pneumonia detection, providing a reliable tool that can assist healthcare professionals in early and accurate diagnosis.

6.2 Limitations

Despite the success achieved in developing an effective pneumonia classification model and integrating it into practical applications, the project has several limitations that need to be acknowledged:

1. **Limited Diagnostic Scope:** The dataset's lack of contextual information, such as the precise location or conditions of sound recordings, may hinder the model's capacity to generalize across varied settings.

2. **Computational Requirements:** EfficientNetB7 is a large model that requires significant computational resources for training and inference. This might limit its deployment in resource-constrained environments.

3. **Limited Testing on Edge Cases:** The model's performance on rare conditions or atypical presentations of pneumonia has not been extensively tested. This could lead to misdiagnosis in uncommon cases.

6.3 Future Scope

The effective implementation of the X-ray image classification project gives up various possibilities for future research and development. Here are potential areas for potential subsequent:

1. **Dataset Enhancement:**

    - Collect a larger and more diverse dataset to improve model generalization.
    - Address class imbalance by augmenting underrepresented classes or collecting more data.

2. **Model Improvements:**

    - Explore other advanced architectures like EfficientNetV2, DenseNet, or custom hybrid models to potentially improve performance.
    - Implement ensemble learning techniques to combine the strengths of multiple models for better accuracy and robustness.

3. **Multi-class Classification:** Develop a multi-class classification model that can identify various lung diseases beyond pneumonia, such as tuberculosis, lung cancer.

4. **Additional Test Data:** Incorporate more diverse test data for more comprehensive test results.

By addressing these areas, the project can significantly advance its impact and applicability in real-world medical settings, ensuring better diagnostic capabilities and improved patient outcomes.

Appendix: Project Setup

1. **Step 1:** Clone the GitHub repository for Android app and flask server:

   ```
   git clone https://github.com/ug932123/X-Ray_image_class_ug.git
   ```

2. **Step 2:** Install the required python packages:

   ```
   pip install -r requirements.txt
   ```

3. **Step 1:** Activate the conda environment where all dependencies are installed for the project.
4. **Step 3:** Run the ngrok tunnel with:

   ```
   ngrok http --domain=your_domain_acquired_via_ngrok 5000
   ```

5. **Step 4:** Run the flask server and make sure port 5000 don't have any prior process running.
6. **Step 5:** Build and deploy the android project on android device.
7. **Step 6:** Start the android application.
8. **Step 7:** Login/Sign up with the credentials and navigate the app as required.

References

[1] Hicham Moujahid, Bouchaib Cherradi, Oussama El Gannour, Lhoussain Bahatti, Oumaima Terrada & Soufiane Hamida (2020). Convolutional Neural Network Based Classification of Patients with Pneumonia using X-ray Lung Images. Advances in Science, Technology and Engineering Systems Journal Vol. 5, No. 5, 167-175. https://dx.doi.org/10.25046/aj050522

[2] Dimpy Varshni, Kartik Thakral, Lucky Agarwal, Rahul Nijhawan & Ankush Mittal (2019). Pneumonia Detection Using CNN based Feature Extraction. 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, 2019, pp. 1-7. https://dx.doi.org/10.1109/ICECCT.2019.8869364

[3] Kishore Medhi, Md. Jamil & Md. Iftekhar Hussain (2020). Automatic Detection of COVID-19 Infection from Chest X-ray using Deep Learning. medRxiv preprint doi: https://doi.org/10.1101/2020.05.10.20097063

[4] Jie Hou1 & Terry Gao (2021). Explainable DCNN based chest X-ray image analysis and classification for COVID-19 pneumonia detection. Sci Rep 11, 16071 (2021). https://doi.org/10.1038/s41598-021-95680-6

[5] Terry Gao, Grace Wang (2020). Chest X-ray image analysis and classification for COVID-19 pneumonia detection using Deep CNN. medRxiv 2020.08.20.20178913; doi: https://doi.org/10.1101/2020.08.20.20178913

[6] Kermany, Daniel; Zhang, Kang; Goldbaum, Michael (2018), Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification. Mendeley Data ,V1, doi: 10.17632/rscbjbr9sj.2, https://data.mendeley.com/datasets/rscbjbr9sj/2

[7] Simon J. D. Prince. Understanding deep learning. The MIT Press, 2023.

[8] Simon S. Haykin. Neural Networks and learning machines. Pearson Education, 2009.

[9] dshahid (2019). Convolutional Neural Network. Towards data science: https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

[10] Dipanjan Sarkar (2018). A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning. Towards data science: https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

[11] dshahid (2019). Introduction to Tensorflow. Medium: https://medium.com/@dshahid380/introduction-to-tensorflow-32d83ece1d64

[12] Mrudul Shah (2019). How to Apply Machine Learning (ML) in an Android App. Towards data science: https://towardsdatascience.com/how-to-apply-machine-learning-ml-in-an-android-app-33e848c0dde6

[13] Furkan Artunç (2020). Serving Machine Learning Models from Python. Towards data science: https://towardsdatascience.com/serving-machine-learning-models-from-python-bd8c43fb61d5

[14] Jimit Dholakia (2020). Creating RESTful Web APIs using Flask and Python. Towards data science: https://towardsdatascience.com/creating-restful-apis-using-flask-and-python-655bad51b24

[15] Flask documentation (2023), The Python Microframework. Flask: https://flask.palletsprojects.com/en/3.0.x/

[16] TensorFlow documentation (n.d), API Documentation. Tensorflow v2.16.1: https://www.tensorflow.org/api_docs

[17] Keras documentation (n.d), Keras 3 API documentation. Keras: https://keras.io/api/

[18] Sklearn documentation (n.d), API Reference. SkLearn: https://scikit-learn.org/stable/api/index.html

[19] Numpy documentation (2023), NumPy Documentation. Numpy v2.0: https://numpy.org/doc/

[20] Ngrok documentation (2024), Overview modules. Ngrok: https://ngrok.com/docs/

[21] Matplotlib documentation (2012-2024), Matplotlib 3.9.0 documentation. Matplotlib: https://matplotlib.org/stable/index.html