

**Dissertation Training**

**At**

**LWE Healthcare**

**Tuberculosis (TB) X-Ray Image Analysis Using Deep  
Learning: Enhancing Diagnostic Accuracy**

**by**

**Mr. RANJEET**

**PG/22/089**

**Under the guidance of  
Dr. Anuradha Bhardwaj**

**PGDM (Hospital & Health Management) 2022-24**





June 15, 2024



**To whomsoever it may concern**

This is to certify that **Mr Ranjeet** , in partial fulfilment of the requirements for the award of the degree of MBA (Hospital and Health Management) from the IIHMR, Delhi has completed his dissertation at **LWE Healthcare** as an **Intern-Operations** during **March to May 2024**.

He has successfully carried out the study designed to his during internship training and his approach to the study has been sincere, scientific, and analytical.  
We wish him all the best for future endeavours.

Mr Yogesh Kumar  
Operations Head  
LWE Healthcare


**TO WHOMSOEVER IT MAY CONCERN**

This is to certify that **Mr. RANJEET** student of **PGDM (Hospital & Health Management)** from the **International Institute of Health Management Research, New Delhi** has undergone internship training at **"LWE Healthcare"** from **March to May 2024**. The Candidate has successfully carried out the study designated to him during the internship training and her approach to the study has been sincere, scientific, and analytical. The Internship is in fulfillment of the course requirements.

I wish him all success in all his future endeavors.



Dr Sumesh Kumar  
Associate Dean, Academic and Student Affairs  
IIHMR, New Delhi


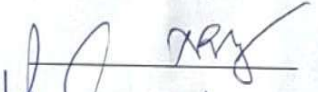



Dr. Anuradha Bhardwaj  
Assistant Professor  
IIHMR, New Delhi

### Certificate of Approval

The following dissertation titled **“Tuberculosis (TB) X-Ray Image Analysis Using Deep Learning: Enhancing Diagnostic Accuracy”** at **“LWE Healthcare”** is hereby approved as a certified study in management carried out and presented in a manner satisfactorily to warrant its acceptance as a prerequisite for the award of PGDM (Hospital & Health Management) for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed, or conclusion drawn therein but approve the dissertation only for the purpose it is submitted.

**Dissertation Examination Committee for evaluation of dissertation.**

Name	Signature
Dr. PRAVEEN KUMAR	
Dr. NISHANT BEH	
Dr. RATNA	

### Certificate from Dissertation Advisory Committee

This is to certify that **Mr. RANJEET**, a graduate student of the PGDM (Hospital & Health Management) has worked under our guidance and supervision. She is submitting this dissertation titled **“Tuberculosis (TB) X-Ray Image Analysis Using Deep Learning: Enhancing Diagnostic Accuracy”** at **“LWE Healthcare”** in partial fulfilment of the requirements for the award of the PGDM (Hospital & Health Management). This Dissertation has the requisite standard and to the best of our knowledge, no part of it has been reproduced from any other dissertation, monograph, report or book.



Dr. Anuradha Bhardwaj  
Assistant Professor  
IIHMR, New Delhi



Organization Mentor  
Yogesh Kumar  
Operations Head  
LWE Healthcare

**INTERNATIONAL INSTITUTE OF HEALTH MANAGEMENT RESEARCH,  
NEW DELHI**

**CERTIFICATE BY SCHOLAR**

This is to certify that the dissertation titled **Tuberculosis (TB) X-Ray Image Analysis Using Deep Learning: Enhancing Diagnostic Accuracy** and submitted by **Mr. RANJEET** Enrolment No. **PG/22/089** under the supervision of **Dr. Anuradha Bhardwaj** for the award of PGDM (Hospital & Health Management) of the Institute carried out during the period from **Feb 2024 to June 2024** embodies my original work and has not formed the basis for the award of any degree, diploma associate ship, fellowship, titles in this or any other Institute or other similar institution of higher learning.



**Signature**

## FEEDBACK FORM

**Name of the Student:** Mr. Ranjeet

**Name of the Organization in Which Dissertation Has Been Completed:** LWE Healthcare

**Area of Dissertation:** Medical Tourism-Operations

**Attendance:** 95%

**Objectives achieved:** Consistently delivered high-quality results and met all the objectives.

**Deliverables:** All tasks were completed on time and with a high degree of accuracy. His work consistently met expectations.

**Strengths:** Maintain a positive attitude and is friendly with colleagues. Professional and strong work ethics.

**Suggestions for Improvement:** Needs to improve ability to delegate tasks to ensure balanced workload. Need to work on long term goals.

**DATE:** 01/07/24  
**PLACE:** New Delhi



**Signature of the Officer-in Charge  
(Internship)**



INTERNATIONAL INSTITUTE OF HEALTH  
MANAGEMENT RESEARCH (IIHMR)

Plot No. 3, Sector 18A, Phase- II, Dwarka, New Delhi- 110075

Ph. +91-11-30418900, [www.iihmrdelhi.edu.in](http://www.iihmrdelhi.edu.in)

CERTIFICATE ON PLAGIARISM CHECK

Name of Student (in block letter)	Dr./Mr./Ms.: <b>RANJEET</b>		
Enrollment/Roll No.	<b>PG/22/089</b>	Batch Year	<b>2022-2024</b>
Course Specialization (Choose one)	Hospital Management	Health Management	Healthcare IT <input checked="" type="checkbox"/>
Name of Guide/Supervisor	Dr./Prof.: <b>ANURADHA BHARDWAJ</b>		
Title of the Dissertation/Summer Assignment	<b>TUBERCULOSIS IMAGE ANALYSIS USING DEEP LEARNING: ENHANCING DIAGNOSTIC ACCURACY.</b>		
Plagiarism detect software used	<b>"TURNITIN"</b>		
Similar contents acceptable (%)	Up to 15 Percent as per policy		
Total words and % of similar contents Identified	<b>13%.</b>		
Date of validation (DD/MM/YYYY)	<b>29/02/2024</b>		

Guide/Supervisor

Name: **DR. ANURADHA BHARDWAS**

Signature: *Anuradha*

Report checked by

Institute Librarian

Signature:

Date:

Library Seal



Student

Name: **RANJEET**

Signature: *Ranjeet*

Dean (Academics and Student Affairs)

Signature:

Date:

(Seal)

**31/7/2024**



## Acknowledgement

This dissertation would not have been possible without the support, guidance, and encouragement of many individuals and institutions.

First and foremost, I would like to express my deepest gratitude to my college mentor, Dr. Anuradha Bhardwaj, for her unwavering support, invaluable guidance, and insightful feedback throughout this research journey. Her expertise and dedication have been instrumental in shaping the direction and quality of this dissertation.

I am also profoundly grateful to the members of my dissertation committee, Dr. Ratika Samtani, Dr. Praveen Kumar, and Dr. Nishikant Bele, for their constructive criticism, suggestions, and encouragement. Their input has been crucial in refining my research and enhancing the overall quality of this work.

A special thank you goes to my organization mentor Mr. Yogesh Kumar, for their guidance, encouragement, and for providing me with the opportunities and resources to balance my professional responsibilities with my academic pursuits. Their support has been critical in allowing me to complete this dissertation.

To my office mates, thank you for your camaraderie, moral support, and insightful discussions. Your encouragement and advice have been a source of motivation and inspiration.

To my family, especially my parents, thank you for your unconditional love, patience, and unwavering support. Your belief in me has been my driving force and I am deeply appreciative of your encouragement throughout this journey.

Thank you all for your invaluable contributions and support.

## Contents

List of Figures .....	11
List of Tables .....	11
List of Abbreviations .....	12
Overview About the Organization .....	13
Abstract .....	14
Introduction.....	15
Literature Review.....	18
Rationale .....	20
Study Objective:.....	20
Methodology .....	21
Results.....	26
Discussion .....	32
Limitations of study .....	34
Conclusion .....	35
Supplementary .....	36
Bibliography .....	67

### List of Figures

S.No.	Header	Pg No.
1	Relationship between the most common AI methods in medicine	16
2	Image enhancement for tuberculosis detecting using deep learning	16
3	Samples of chest X-ray images	20
4	ConvNet Architecture	21
5	ResNet-50 Architecture	22
6	VGG16 Architecture	22
7	Evaluation criteria	24
8	VGG16 Confusion matrix	26
9	VGG16 Classification report	26
10	VGG16 ROC curve	26
11	ResNet50 Confusion matrix	27
12	ResNet50 Classification report	27
13	ResNet50 Classification report	27
14	CNN Confusion matrix	28
15	CNN Classification report	28
16	CNN Classification report	28

### List of Tables

S.No.	Header	Pg No.
1	Summary of tuberculosis (TB) chest X-ray datasets	20
2	Tools used in the study	23
3	Model performance on the test set	25

## List of Abbreviations

1. TB – Tuberculosis
2. DICOM – Digital Imaging and Communications in Medicine
3. DR – Digital Radiography
4. TP – True Positive
5. FP – False Positive
6. TN – True Negative
7. OS – Operating System
8. CV2 – Computer vision 2
9. CXR– Chest X-Ray
10. CNN – Convolutional Neural Network
11. AI – Artificial Intelligence
12. WHO – World Health Organization
13. VGG16 – Visual Geometry Group 16
14. CAD – Computer Aided Diagnostic
15. DCNN – Deep Convolutional Neural Networks
16. ICMR – Indian Council of Medical Research

## Overview About the Organization



LWE Healthcare – A Unit of LWE CARE PHARMACEUTICALS PVT LTD is a leading medical tourism company headquartered in India, dedicated to providing seamless healthcare solutions to global patients seeking affordable and world-class medical treatments. With a strong network of renowned hospitals, experienced physicians, and state-of-the-art facilities across the country, we ensure that every patient receives personalized care and attention throughout their medical journey.

Our mission is to bridge the gap between quality healthcare and international patients by offering comprehensive services, including medical consultations, treatment planning, travel arrangements, and accommodation. Committed to excellence, we prioritize patient safety and satisfaction, adhering to the highest standards of medical care and ethical practices. At LWE Patient Care, we strive to make India a preferred destination for medical tourists, delivering exceptional healthcare experiences and successful treatment outcomes.

## Abstract

### **Purpose**

The primary objective of this study was to develop and validate Deep learning models for the accurate processing of tuberculosis (TB) X-ray images to increase diagnosis accuracy and efficiency in clinical settings. The study aimed to enhance early tuberculosis detection rates by utilizing deep learning techniques to mitigate the subjectivity and unpredictability associated with traditional chest radiograph (CXR) interpretation.

### **Materials & Methods**

A dataset of postero-anterior chest radiographs from the Indian Council of Medical Research (ICMR) TB portal was made publicly available for use in the study. There were 3782 TB cases and 3960 healthy cases in the dataset. To improve the dataset, the images were preprocessed and enhanced. Three convolutional neural network (CNN) models were created and assessed: VGG16, ResNet50, and a CNN that was trained from scratch. Accuracy, precision, recall, F1 score, and area under the curve (AUC) were the metrics used to evaluate these models.

### **Results**

In comparison to the pre-trained models, the CNN model that was trained from scratch performed better, attaining an accuracy of 80%, precision and recall of 0.80, and an F1 score of 0.80 for TB cases. By contrast, VGG16 was able to obtain 52% accuracy for TB cases, with precision of 0.44, recall of 0.28, and F1 score of 0.35. ResNet50 yielded an F1 score of 0.72 for TB cases, accuracy of 69%, precision of 0.65, recall of 0.81, and performance greater than VGG16 but still below the custom CNN.

### **Conclusion**

The study showed how personalized CNN models could enhance the precision of tuberculosis diagnosis using chest radiographs. The CNN that was trained from scratch outperformed pre-trained models, demonstrating the value of customizing model training for medical imaging applications. To encourage the use of AI-driven diagnostic tools in healthcare, future research should concentrate on growing the dataset, including clinical data, and improving model interpretability. The results highlight how AI and Deep learning have the potential to revolutionize medical diagnostics and open the door to more accurate, effective, and affordable healthcare options.

## PROJECT REPORT

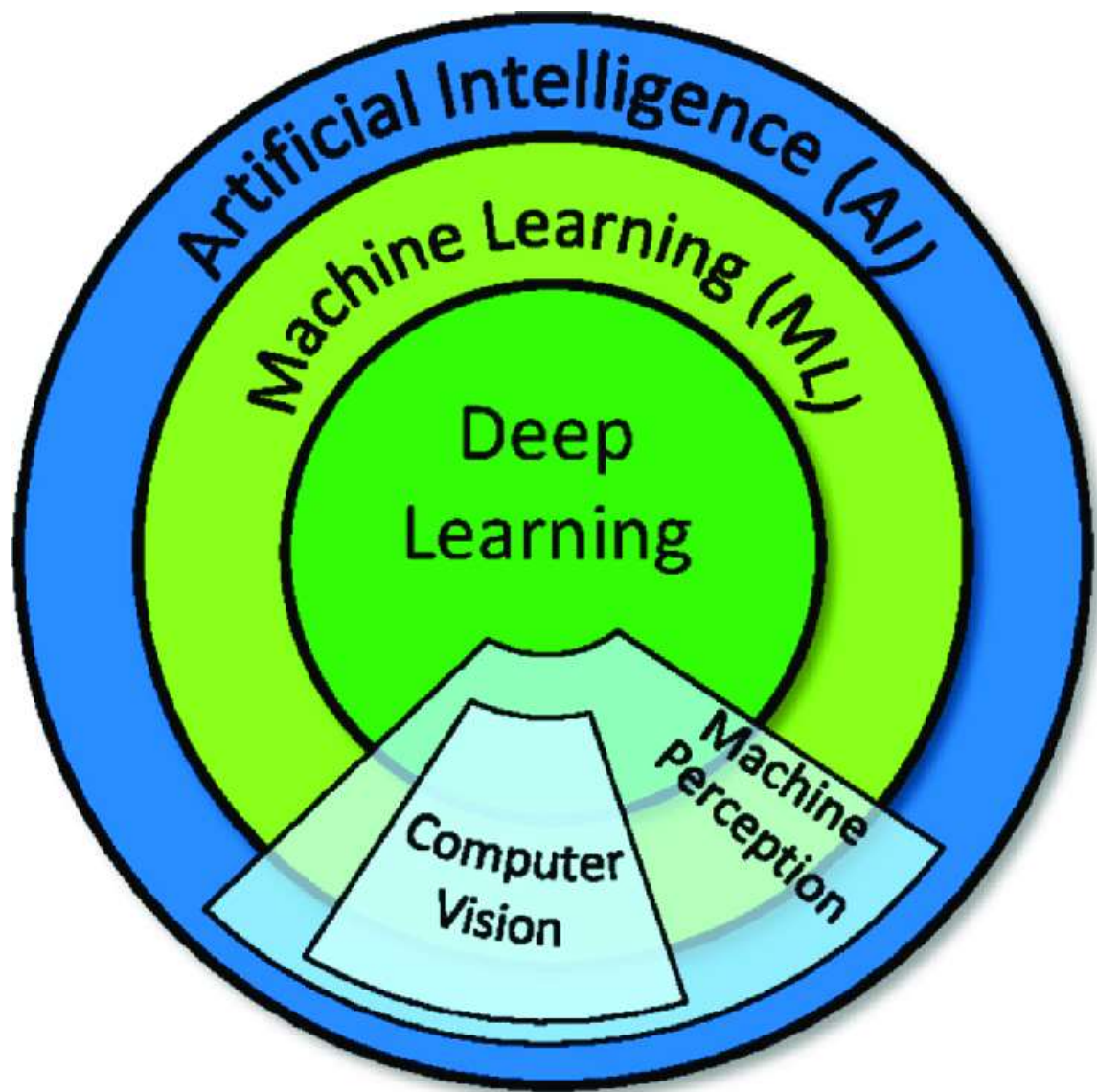
# **Tuberculosis (TB) X-Ray Image Analysis Using Deep Learning: Enhancing Diagnostic Accuracy**

### Introduction

Lung disease known as tuberculosis (TB) is brought on by bacterial infection. The bacillus *Mycobacterium tuberculosis* is the type of bacteria that causes tuberculosis. This infectious disease that spreads through the air is ranked among the top 10 global causes of death. Several diagnostic procedures will be necessary to discover tuberculosis (TB), as it is rather difficult to diagnose in its early stages compared to other infectious diseases. To provide early TB diagnosis, the World Health Organisation (WHO) advises widespread use of a systematic screening approach. Because of its relatively high sensitivity, despite the low specificity and significant intra- and inter-observer variability of chest X-rays (CXR), the World Health Organization (WHO) endorses CXR as one of the primary modalities for TB detection and screening.

However, as previously mentioned, there are notable differences in the interpretation of CXR across and among observers, leading to incorrect TB diagnoses. CXR interpretation is also a laborious and subjective procedure. Furthermore, TB's radiologic patterns resemble those of other lung conditions, which may cause a false positive. For this reason, computer-aided diagnostic (CAD) systems have been developed recently that can identify tuberculosis (TB) automatically from chest radiography. Using chest radiography, CAD solutions use picture segmentation, texture and form feature extraction, and classification techniques to diagnose pulmonary tuberculosis.

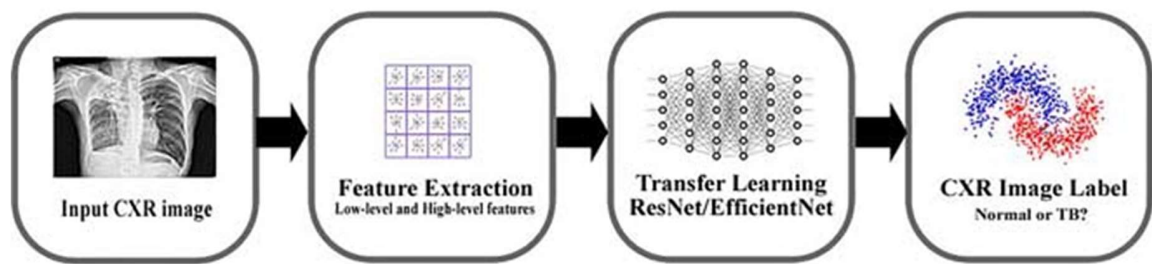
Throughout the past ten years, artificial intelligence (AI)-based systems have been employed for physiological monitoring, brain tumor and breast cancer diagnosis, among other applications. Computers can learn on their own without explicit programming thanks to a branch of artificial intelligence called machine learning, or self-learning. Stated differently, machine learning recognizes patterns in data (images, for example). Deep learning is a kind of machine learning that takes raw data and employs numerous layers to extract higher-level features. Deep learning algorithms have become the most sophisticated methods for classifying images in recent years. Deep convolutional neural networks, or CNNs, are one of the more intriguing deep learning methods for photo classification.



*Figure 1 Relationship between the most common AI methods in medicine*

CNNs have recently been employed in several studies to automatically diagnose lung conditions like pneumonia from CXR. Using pre-trained models and their ensembles, concept transfer learning is used in deep learning frameworks to identify tuberculosis. Our goal was to automatically identify tuberculosis (TB) from CXR pictures using a CNN model that I trained from scratch. I then compared the CNN model's performance to three different pre-trained CNNs using a transfer learning technique.





*Figure 2 Image enhancement for tuberculosis detecting using deep learning*

The glossary of terminology provided here may aid in comprehending the operation of deep learning.

**Classification** is the process of applying a class or label to a group of pixels, such as those labelled as tumours, using a segmentation method. After segmenting and labelling a section of an image as "abnormal brain," for instance, the classifier might try to determine if the highlighted area represents malignant or benign tissue.

**Algorithm:** An algorithm is a collection of steps used to construct a model that will be used to most accurately predict classes based on the properties of the training samples.

**Labelled data** is a set of examples (images, for example) that have been assigned a specific "answer." The specific location of a tumor may be the solution for some occupations; for other duties, it may depend on the type of cancer the lesion represents or whether cancer is present at all.

**Training:** The phase in which the deep learning algorithm system receives labeled example data that contains the responses, or labels.

**Validation:** The validation set is the collection of examples that were utilized in the training process. This is also known as the training set.

**Testing:** Using a third set of examples, "real-world" testing is sometimes carried out. Since the algorithm system can iterate to improve performance using the validation set, it is possible that it will learn characteristics from the training set. A high level of performance on a "unseen" test set can increase the probability that the algorithm will yield reliable answers in real-world scenarios.

## Literature Review

Study	Objective	Data Sets	Methods/Techniques Used	Key Findings	Accuracy (%)
Liu et al.	Diagnosis of tuberculosis using chest radiographs	Three distinct data sets	Neural Network (NN)	High accuracy in TB diagnosis	89.0 - 96.1
Khan et al.	Classification of patient records for TB diagnosis	12,600 patient records	Neural Network (NN)	Effective distinction between positive and negative TB	>94
Ghanshal et al.	Identifying tuberculosis using various ML techniques	Not specified	SVM, kNN, RF, NN	NN outperformed other classifiers in TB detection	80.45
Chandra et al.	Automated identification of abnormal CXR images	Two public data sets	SVM with hierarchical feature extraction	High accuracy in detecting TB-related diseases	95.6 - 99.4
Lakhani and Sundaram	TB identification on chest radiographs using deep CNNs	Not specified	Ensemble of GoogLeNet and AlexNet	Highly effective in TB detection	98.7
Hooda et al.	Ensemble DL-based TB detection system	Four distinct data sets	AlexNet, GoogLeNet, ResNet	Ensemble approach demonstrated good performance	88.24
Heo et al.	TB identification on chest	Yearly worker health	VGG-19, InceptionV3, ResNet-50,	Various CNN architecture	Not specified but various

	radiographs using DL techniques	examination data	DenseNet-121, InceptionResNetV2	s used, one integrated with demographic information	architectures evaluated
--	---------------------------------------	---------------------	------------------------------------	---	----------------------------

## Rationale

To use a transfer learning-based approach to automatically diagnose tuberculosis (TB) from chest X-ray (CXR) pictures by training an entirely new convolutional neural network (CNN) model and comparing its performance with two other pre-trained CNNs, namely VGG16 & ResNet50.

## Study Objective:

### **General Objective:**

To develop and validate Deep learning models for the accurate analysis of Tuberculosis (TB) X-ray images, enhancing diagnostic precision and efficiency in clinical settings.

### **Specific Objectives:**

- 1.To collect and preprocess a comprehensive dataset of Tuberculosis (TB) X-ray images.
2. To design and train multiple Deep Learning models trained for Tuberculosis (TB) X-ray image analysis.
- 3.To apply common metrics to assess these models' success.

## Methodology

### Dataset

The datasets of postero-anterior chest radiographs from the Indian Council of Medical Research (ICMR) TB portal, which are publicly available, were used in this study (Table 1). Sample instances from both datasets—normal and TB—are displayed in Figure 3. Public access to both datasets can be found here: <https://nirt.res.in/html/xray.html>

Dataset	No. of healthy cases	No. of TB cases	File type	Radiology system
ICMR TB portal	3960	3782	DICOM	DR

Table 1: An overview of the datasets for tuberculosis (TB) chest X-rays



Figure 3 Samples of chest X-ray images

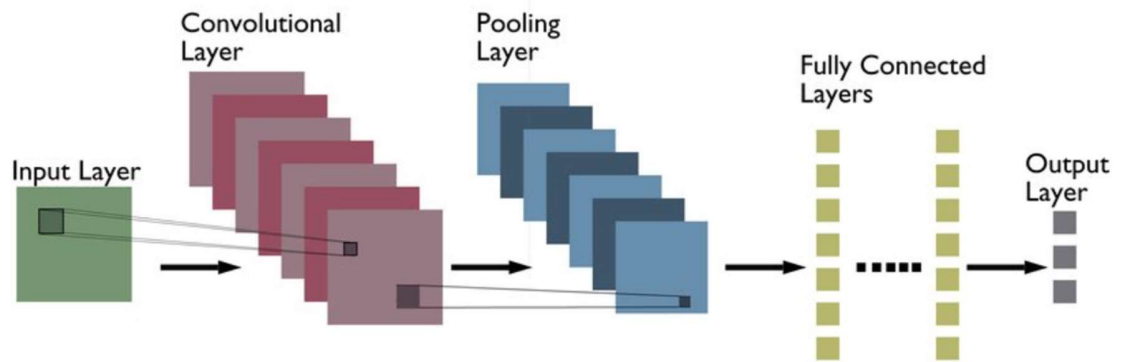
### Data preprocessing and augmentation

Because the input images varied in size, the CXR images in this investigation were scaled to  $256 \times 256$  pixels for my trained model and  $224 \times 224$  pixels for VGG16 and ResNet50. Input images were converted from DICOM to JPEF format. Next, methods for augmenting data were used. According to reports, deep learning systems' classification accuracy can be increased by employing data augmentation. Moreover, data augmentation can greatly expand the sample sizes in datasets used to train models. Here, the picture can be improved by filliping on the horizontal and vertical axes, a rotation range of 10, a width shift range of 0.1, a height shift range of 0.1, and a zoom range of 0.1. Following data augmentation, 6465

photos were classified into 48.64% TB and 51.35% normal categories during the training phase. There are 1275 photographs for the testing phase 49.8% TB and 50.19% normal.

### **Proposed model**

The convolution layer is denoted by the name "Conv" in the convolutional neural network (CNN or ConvNet) that I trained from scratch in this work.



*Figure 4 ConvNet Architecture*

**Convolutional layers** are thought to be the fundamental CNN building component. Convolution is used by CNNs in place of standard matrix multiplication. A group of filters referred to as convolutional kernels make up convolutional layers. Using kernels, the convolutional layer's primary function is to extract certain features from an input image. Rectified Linear Units, or ReLUs, are typically utilized as deep learning activation layers.

**Layer of pooling** to reduce the spatial dimension of the input data and hence the number of network parameters, an optional pooling or down sampling layer is added in CNN after the convolutional layer. The most popular method of pooling is known as pooling. In addition, two further pooling strategies are L2-norm pooling and average pooling.

**Fully connected layer:** Every neurone in one layer is connected to every other neurone in the layer below through fully connected layers. The fully connected layer receives its input as the flattened output of the last pooling or convolutional layer.

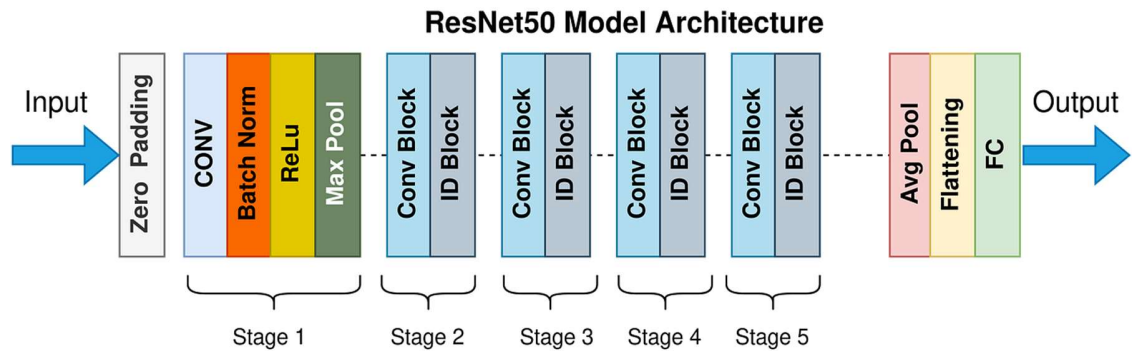
### **Pre-trained transfer models**

In this case, the datasets with TB CXR images were utilized. After partitioning and pre-processing the dataset, picture augmentation techniques were applied for the training phase. Over-fitting is avoided because of the data augmentation.

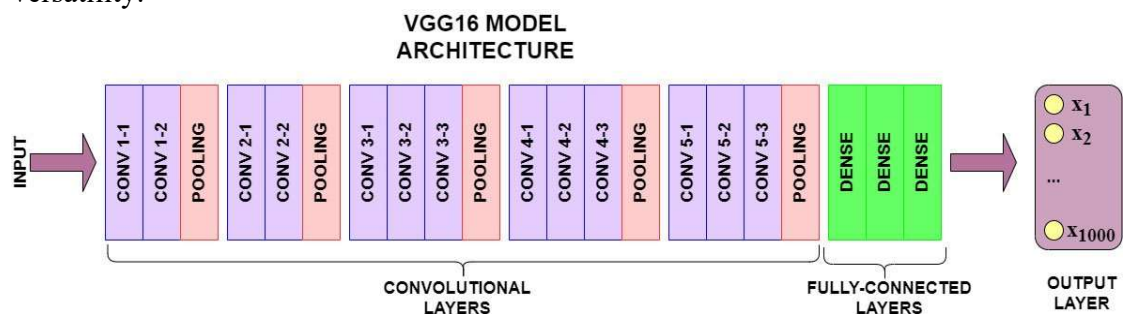
A CNN-based algorithm was applied to identify tuberculosis using CXR images. In the current work, two different pre-trained models were used in a CNN-based transfer learning technique to classify CXR photos into normal and TB (binary classification).

- **ResNet50:** ResNet-50 is a CNN design that belongs to the ResNet (Residual Networks) family of models, which was developed to address the challenges associated with training deep neural networks. One well-known deep learning

model for picture categorization tasks is called ResNet-50, which was created by researchers at Microsoft Research Asia. There are several depths of ResNet architectures available, such as ResNet-18, ResNet-32, and so on. A mid-sized variant of the architecture is called ResNet-50. In the history of picture categorization, ResNet-50 is still recognized as a noteworthy model, even though it was introduced in 2015.



- VGG16:** The University of Oxford's Visual Geometry Group (VGG) presented the VGG-16 model, a convolutional neural network (CNN) architecture. With a total of 16 layers—13 convolutional layers and 3 fully connected layers—it stands out for its depth. VGG-16 is well known for its high performance on a variety of computer vision tasks, such as object identification and picture categorization, in addition to its efficacy and convenience of use. The model is built with a stack of progressively deeper max-pooling layers stacked on top of a series of convolutional layers. One aspect of the model that helps it provide reliable and accurate predictions is its capacity to learn complex hierarchical representations of visual data. Despite being simpler than more modern architectures, VGG-16 is still a popular choice for many deep learning applications because of its exceptional speed and versatility.



Put another way, a convolution layer, a pooling layer, a flattening layer, and a fully linked layer make up the common architecture of all pre-trained transfer models. The fully linked layer in this instance is made up of the following layers in order to forecast

the normal and TB cases (binary classification):

- Slatched
- dense, consisting of 256 units
- dropout with a 0.2 threshold.
- a final dense with the SoftMax activation of two elements.

### **Training phase**

To decrease the dimension of the retrieved features, I trained the suggested models using the Adam optimizer and the categorical cross-entropy loss function. The following solver parameters were used for training: 50 batch sizes, 0.00001 learning rate, and 200 epoch values. As was previously indicated, techniques for data augmentation have been applied to reduce overfitting and boost training effectiveness. Using the Python library, this effort trained, verified, and tested a large number of algorithms. The Holdout technique was used to assess the suggested models' performance in binary classification.

Environment	Anaconda
IDE	Jupyter Notebook
Language	Python
Libraries	NumPy, OpenCV-python, TensorFlow, Matplotlib, Seaborn, Skit-learn, Pydicom, Pillow, Glob, OS, CV2

Table 2: Tools used in the study

### **Evaluation criteria**

Five performance measures are used to assess and compare the effectiveness of the various suggested models for the testing dataset. Equations for the parameters are given below. The parameters are:

- Accuracy (1)
- Sensitivity/Recall (2)
- Precision (3)
- Area under curve (AUC) (4)
- F1 score (5)



$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{FP} + \text{TN} + \text{FN})} \quad (1)$$

$$\text{Recall} = \frac{(\text{TP})}{(\text{TP} + \text{FN})} \quad (2)$$

$$\text{Precision} = \frac{(\text{TP})}{(\text{TP} + \text{FP})} \quad (3)$$

$$\text{F1 - score} = 2 \times \frac{(\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})} \quad (4)$$

*Figure 7 Evaluation criteria*

## Results

Tuberculosis				
Model	Precision	Recall	F1 score	Accuracy
VGG 16	0.44	0.28	0.35	0.52
RESNET50	0.65	0.81	0.72	0.69
CNN	0.80	0.80	0.80	0.80

Normal				
Model	Precision	Recall	F1 score	Accuracy
VGG 16	0.55	0.71	0.62	0.52
RESNET50	0.75	0.57	0.65	0.69
CNN	0.80	0.80	0.80	0.80

Table 3: Model performance on the test set

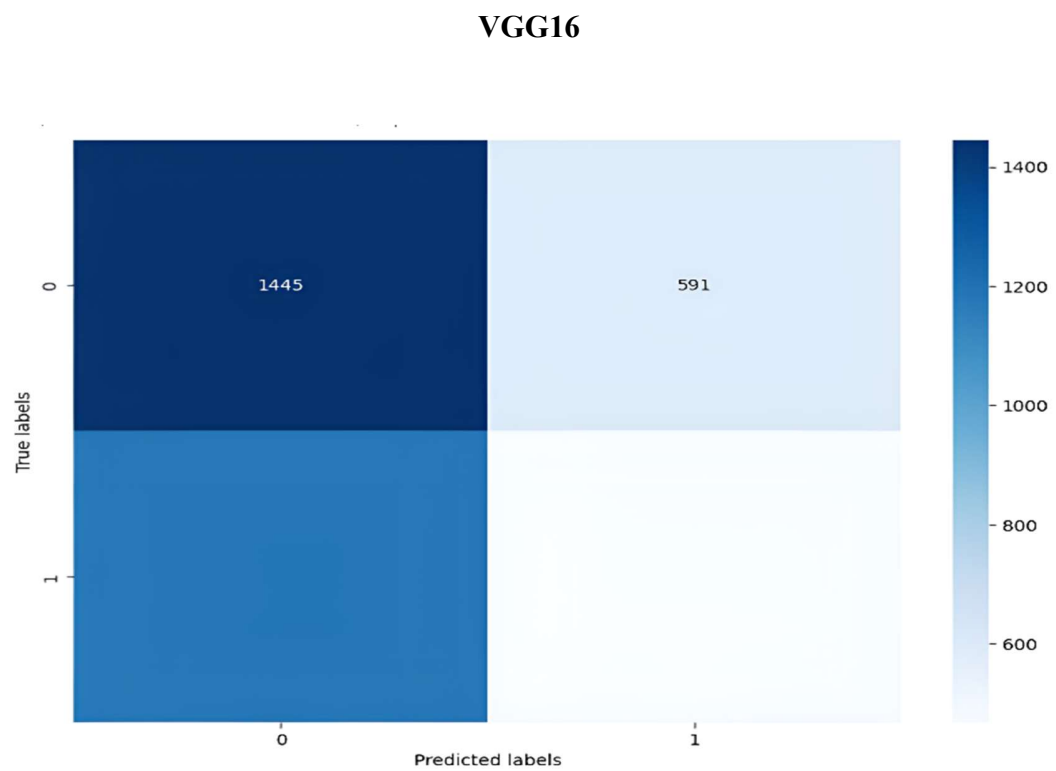


Figure 8 VGG16 Confusion matrix

	precision	recall	f1-score	support
Normal	0.55	0.71	0.62	2036
Tuberculosis	0.44	0.28	0.35	1646
accuracy			0.52	3682
macro avg	0.50	0.50	0.48	3682
weighted avg	0.50	0.52	0.50	3682

Figure 9 VGG16 Classification report

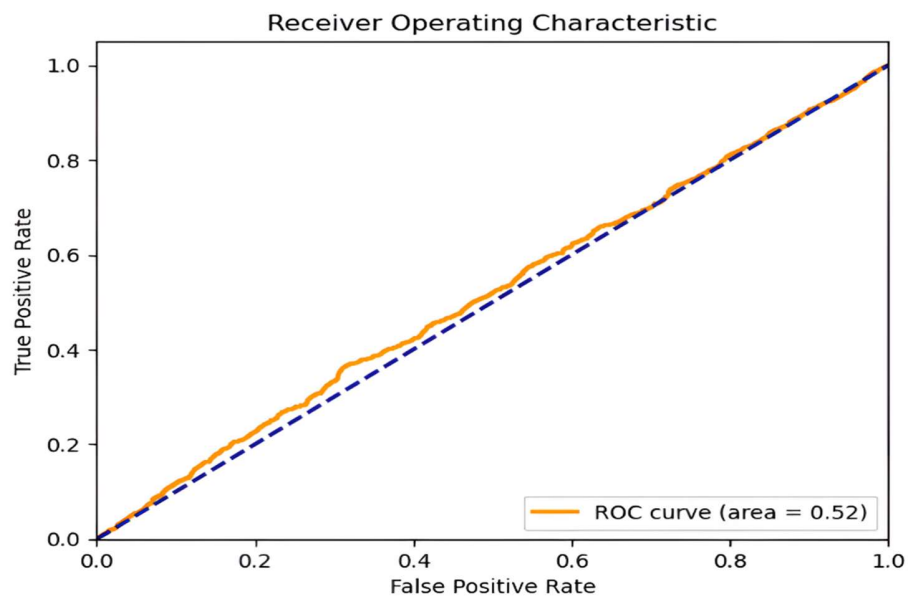


Figure 10 VGG16 ROC curve

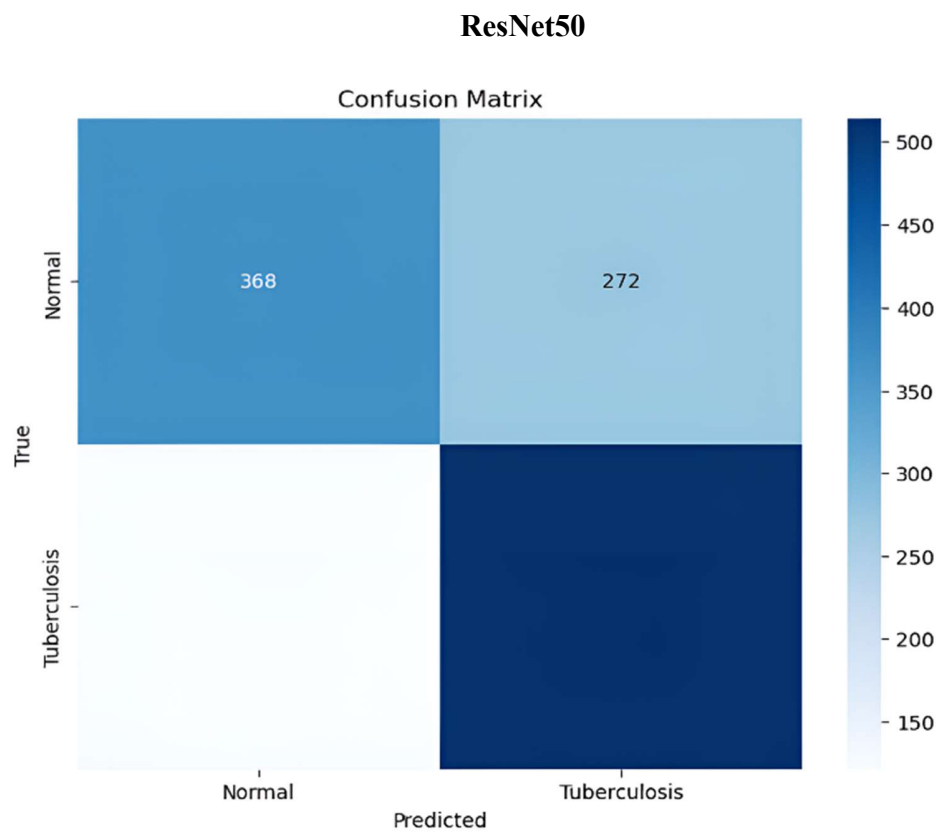
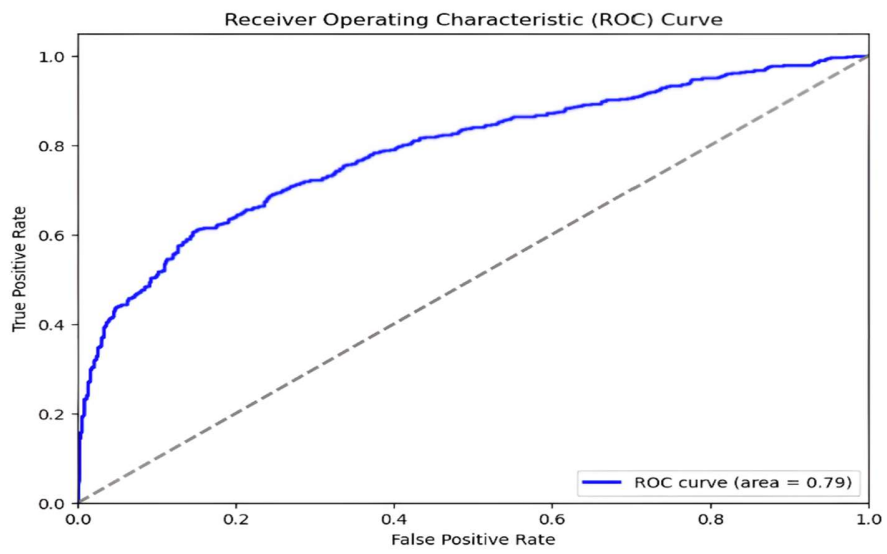


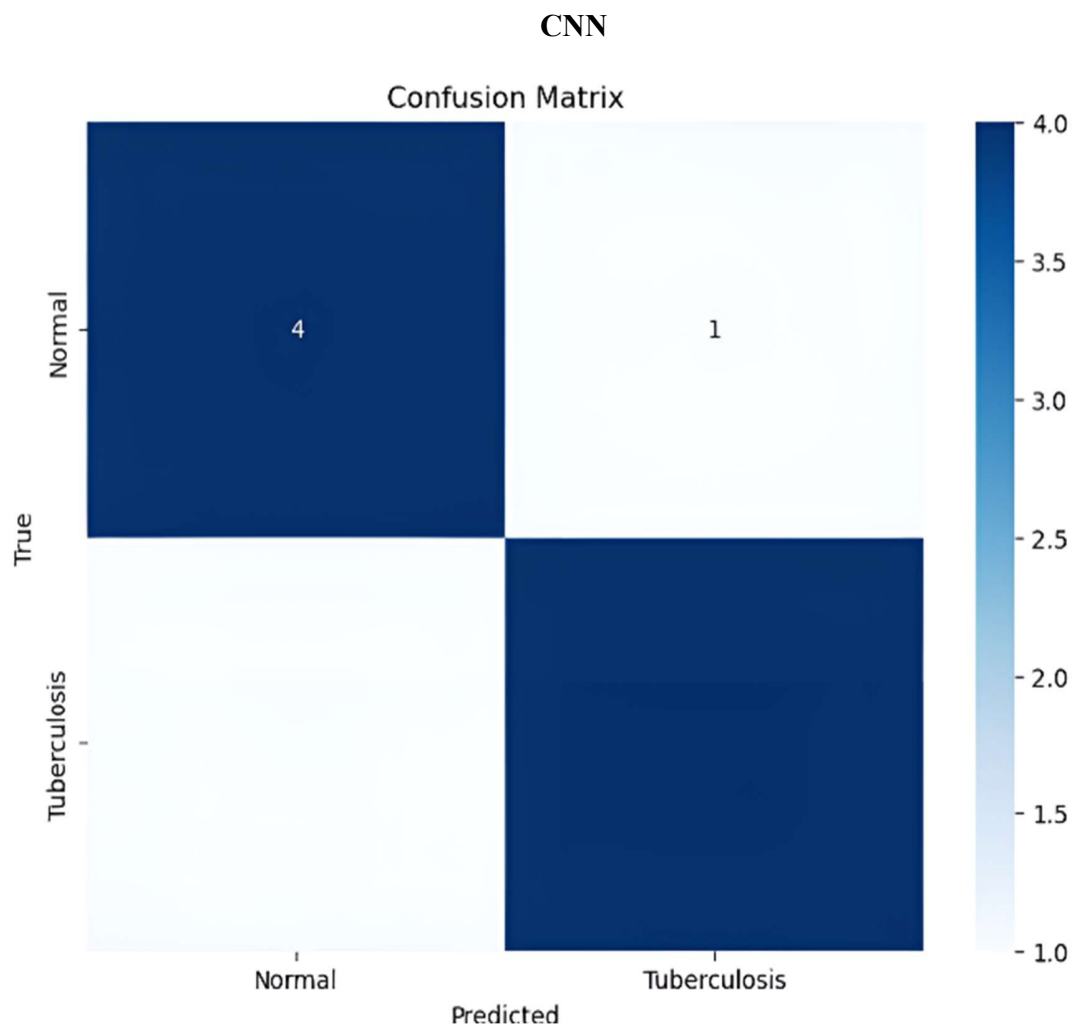
Figure 11 ResNet50 Confusion matrix

	precision	recall	f1-score	support
Normal	0.75	0.57	0.65	640
Tuberculosis	0.65	0.81	0.72	635
accuracy			0.69	1275
macro avg	0.70	0.69	0.69	1275
weighted avg	0.70	0.69	0.69	1275

*Figure 12 ResNet50 Classification report*



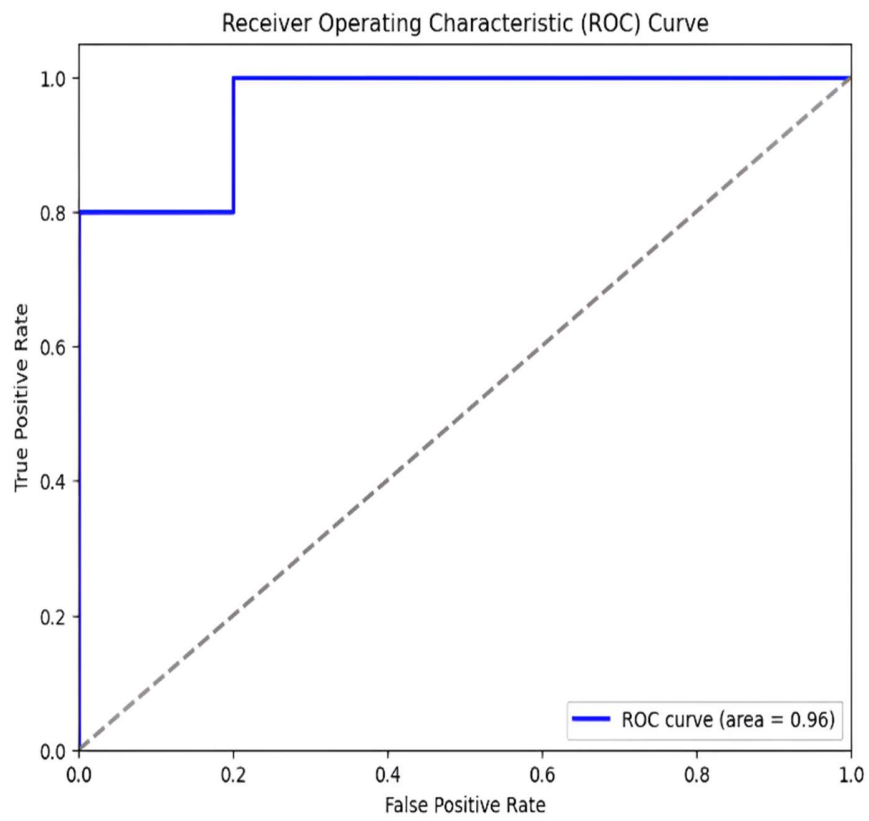
*Figure 13 ResNet50 ROC curve*



*Figure 14 CNN Confusion matrix*

	precision	recall	f1-score	support
Normal	0.80	0.80	0.80	5
Tuberculosis	0.80	0.80	0.80	5
accuracy			0.80	10
macro avg	0.80	0.80	0.80	10
weighted avg	0.80	0.80	0.80	10

*Figure 15 CNN Classification report*



*Figure 16 CNN ROC Curve*

## Discussion

The performance of different convolutional neural network (CNN) models in diagnosing tuberculosis (TB) using chest X-ray (CXR) pictures was carefully assessed. These models included VGG16, ResNet50, and a CNN trained from scratch. The comparative research highlights the significance of model selection and training approaches in clinical AI applications by revealing significant differences in the models' diagnostic accuracy, precision, recall, and F1 scores.

### CNN Trained from Scratch

Developing a CNN model from scratch yielded the best overall performance, with an 80% accuracy rate. Because the TB CXR dataset was used exclusively to train the algorithm, it was able to identify and understand patterns specific to TB symptoms in radiographs. With an F1 score of 0.80, the model's recall and precision were both 0.80. These measures show a strong capacity to minimise the frequency of false positives while accurately identifying TB cases (true positives).

- Accuracy: 80%
- Precision: 0.80
- Recall: 0.80
- F1 Score: 0.80

Excellent recall and precision ratings are very important when diagnosing tuberculosis. To guarantee that a significant percentage of cases of tuberculosis are in fact tuberculosis, precision measures the percentage of true positive diagnoses among all positive diagnoses produced by the model. Recall shows the percentage of real TB cases that the model accurately recognized, indicating its sensitivity and dependability in TB case detection.

### Pre-trained Models: VGG16 and ResNet50

On the other hand, the pre-trained models ResNet50 and VGG16 showed worse performance metrics. Even though VGG16 is a well-known architecture for picture classification, its accuracy was just 52%. With recall values of 0.71 and 0.28, its precision decreased to 0.44 for tuberculosis cases from 0.55 for normal cases. These numbers led to F1 scores for TB cases of 0.35 and normal cases of 0.62.

- Accuracy: 52%
- Precision (Normal): 0.55
- Recall (Normal): 0.71
- F1 Score (Normal): 0.62
- Precision (TB): 0.44
- Recall (TB): 0.28
- F1 Score (TB): 0.35



While ResNet50, another well-known model, outperformed VGG16, it was still not as good as the CNN that was trained from scratch. With 69% accuracy overall, ResNet50 performed well. Its F1 score was 0.65 based on precision and recall of 0.75 and 0.57 for typical situations, respectively. The precision and recall for TB cases were 0.65 and 0.81, respectively, yielding an F1 score of 0.72.

- Accuracy: 69%
- Precision (Normal): 0.75
- Recall (Normal): 0.57
- F1 Score (Normal): 0.65
- Precision (TB): 0.65
- Recall (TB): 0.81
- F1 Score (TB): 0.72

#### Comparative Analysis and Implications

The CNN model that was created from scratch performed better than the others because it was specifically trained on the TB dataset. The customized CNN model was tailored for the subtleties and unique characteristics of TB radiographs, in contrast to pre-trained models, which are trained on a variety of pictures for broad image classification tasks. Better feature extraction and pattern recognition relevant to tuberculosis diagnosis were made possible by this customized method.

Several important points are highlighted by the performance metrics:

- **Balanced Precision and Recall:** In medical diagnostics, where false positives and false negatives can have serious repercussions, the CNN trained from scratch was able to retain a balanced precision and recall. If a case has a high recall, then most TB cases will be found, and if it has a high precision, then most cases found will be TB cases.
- **Restrictions on Pre-trained Models:** Even though VGG16 and ResNet50 have strong architectures, their poorer performance in this job highlights the drawbacks of using pre-trained, general models for specialized medical imaging tasks without further modification and fine-tuning.
- **Training:** Diagnostic accuracy is greatly improved by customizing the training procedure to the medical imaging task and dataset. This result validates the expenditure on creating and refining unique models for certain medical use cases.

### Limitations of study

- **Lack of Diversity:** Limited representation of different demographics and TB manifestations affects generalizability.
- **Model Complexity:** Deep CNNs are complex and hard to interpret, posing challenges in medical contexts.
- **Overfitting Risk:** Training from scratch on a small dataset can cause the model to overfit.
- **Hyperparameter Tuning:** Improper tuning can lead to suboptimal results and requires significant resources.
- **Lack of External Validation:** Absence of validation with an independent dataset reduces confidence in model robustness.
- **Absence of Clinical Validation:** No comparison against clinical diagnoses or outcomes to ensure practical applicability.
- **Domain Adaptation:** Pre-trained models may not capture disease-specific features relevant to TB.
- **Computational Resources:** High resource requirement for training deep models from scratch.

## Conclusion

Our research uses deep CNNs in combination with transfer learning to automatically identify TB cases from normal cases based on chest radiographs. For TB CXR image detection, the efficacy of three distinct CNN models was assessed. For the datasets using image augmentation techniques, With an accuracy of 80%, the CNN model that was trained from scratch outperformed the VGG16 and ResNet50 models that were pre-trained, which had accuracies of 52% and 69%, respectively. Additionally, the customized CNN model kept a balanced recall and precision of 0.80, demonstrating strong diagnostic skills with low false negatives and positives. The accuracy and robustness of our suggested models can be increased by using more datasets; future research must address this issue.

Even with the encouraging outcomes, a number of difficulties still exist. These include the possibility of overfitting, the requirement for more extensive and varied datasets, and the significance of creating AI models that are comprehensible in order to win over doctors. For AI-driven diagnostic technologies to be successfully integrated into healthcare, several issues must be resolved.

Subsequent investigations ought to concentrate on broadening the dataset, merging imaging and clinical data, and improving model interpretability. Additionally, to verify the models' effectiveness in practical contexts, outside validation using separate datasets and clinical trials will be required.

The study's findings highlight the revolutionary possibilities of AI and Deep learning in the field of medical diagnostics. A noteworthy development in the use of technology to enhance the provision and results of healthcare is the creation and implementation of customized CNN models for tuberculosis diagnosis. We can get closer to achieving AI's full potential in medicine by further improving and validating these models, which will ultimately result in more precise, effective, and easily accessible healthcare solutions.

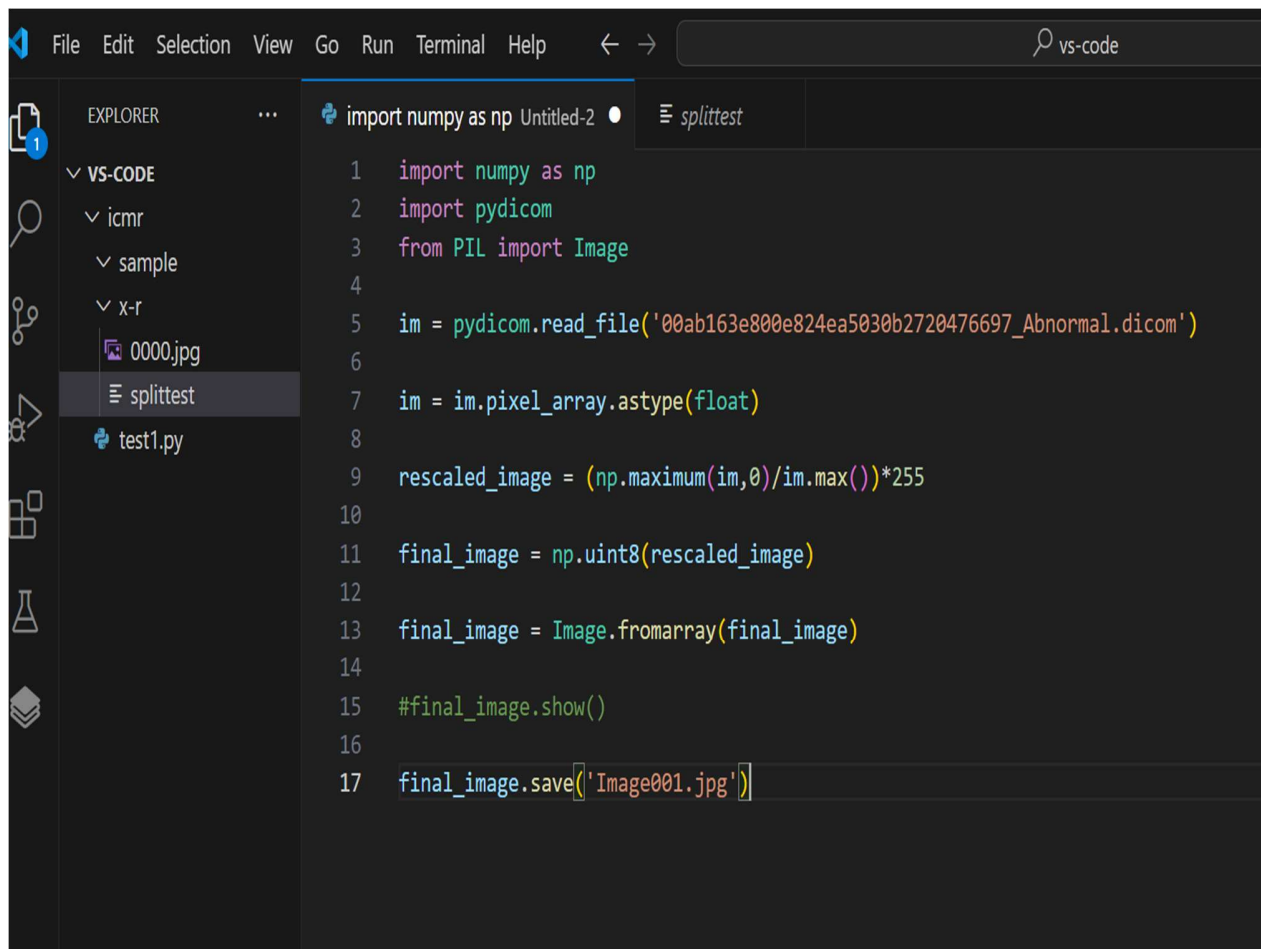
## Supplementary

### Instrumentation

- Computer programming code

### Code for Convert Dicom image format to jpg image format

#### For Single Image



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

im = pydicom.read_file('00ab163e800e824ea5030b2720476697_Abnormal.dicom')
im = im.pixel_array.astype(float)
rescaled_image = (np.maximum(im,0)/im.max())*255
final_image = np.uint8(rescaled_image)
final_image = Image.fromarray(final_image)
#final_image.show()
final_image.save('Image001.jpg')
```

The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a file tree with 'VS-CODE', 'icmr', 'sample', 'x-r', '0000.jpg', 'splittest', and 'test1.py'. The 'splittest' file is selected. The main editor area shows a Python script for converting a DICOM image to a JPG format. The script imports numpy, pydicom, and PIL Image. It reads a DICOM file, converts the pixel array to float, rescales it to a range of 0-255, converts it back to uint8, and saves it as a JPG file. A commented line shows how to display the image.

#### For multiple Images

```
1  import os
2  import numpy as np
3  import pydicom
4  from PIL import Image
5
6
7  input_dir = 'C:/Users/Ranjeet/Data/ICMR'
8  output_dir = 'C:/Users/Ranjeet/Data/ICMR/x-ray'
9
10
11  os.makedirs(output_dir, exist_ok=True)
12
13  for filename in os.listdir(input_dir):
14      if filename.endswith('.dicom'):
15
16          filepath = os.path.join(input_dir, filename)
17
18
19          im = pydicom.read_file(filepath)
20
21
22          im = im.pixel_array.astype(float)
23
24
25          rescaled_image = (np.maximum(im, 0) / im.max()) * 255
26
27
28          final_image = np.uint8(rescaled_image)
29
30
31          final_image = Image.fromarray(final_image)
32
33
34          output_filepath = os.path.join(output_dir, f'{os.path.splitext(filename)[0]}.jpg')
35          final_image.save(output_filepath)
36
37          print(f'Saved {output_filepath}')
38
```

## X-ray Image classification using VGG16 architecture.

```
•[15]: import warnings
warnings.filterwarnings('ignore')

# Import necessary libraries
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import load_model
import numpy as np
from glob import glob
import matplotlib.pyplot as plt

[40]: # Define image size
IMAGE_SIZE = [224, 224]

# Define paths
train_path = 'Data/ICMR/train_'
valid_path = 'Data/ICMR/test_'

[ ]:

[17]: # Load VGG16 model with pre-trained ImageNet weights
vgg = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)

# Freeze all the layers
for layer in vgg.layers:
    layer.trainable = False

[21]: # Get number of classes
folders = glob(train_path + '/*')

# Add custom layers
x = Flatten()(vgg.output)
prediction = Dense(len(folders), activation='softmax')(x)
```



```
# Create model object
model = Model(inputs=vgg.input, outputs=prediction)

# View model structure
model.summary()
```

Model: "functional\_1"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool1 (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool1 (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool1 (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool1 (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool1 (MaxPooling2D)	(None, 7, 7, 512)	0

block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 3)	75,267

Total params: 14,789,955 (56.42 MB)

Trainable params: 75,267 (294.01 KB)

Non-trainable params: 14,714,688 (56.13 MB)

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

```
[27]: # Data augmentation for training data
train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)

# Data augmentation for validation data
test_datagen = ImageDataGenerator(rescale=1./255)

# Load training data
training_set = train_datagen.flow_from_directory(train_path, target_size=(224, 224), batch_size=10, class_mode='categorical')

# Load validation data
test_set = test_datagen.flow_from_directory(valid_path, target_size=(224, 224), batch_size=10, class_mode='categorical')
```

Found 6465 images belonging to 3 classes.

Found 1275 images belonging to 2 classes.

```
[29]: test_set.class_indices
```

```
[29]: {'Normal': 0, 'Tuberculosis': 1}
```



```
jupyter new1 Last Checkpoint: last month
File Edit View Run Kernel Settings Help Trusted
JupyterLab Python 3 (ipykernel)

[ ]: # Train the model
r = model.fit(
    training_set,
    validation_data=test_set,
    epochs=10,
    steps_per_epoch=len(training_set),
    validation_steps=len(test_set)
)

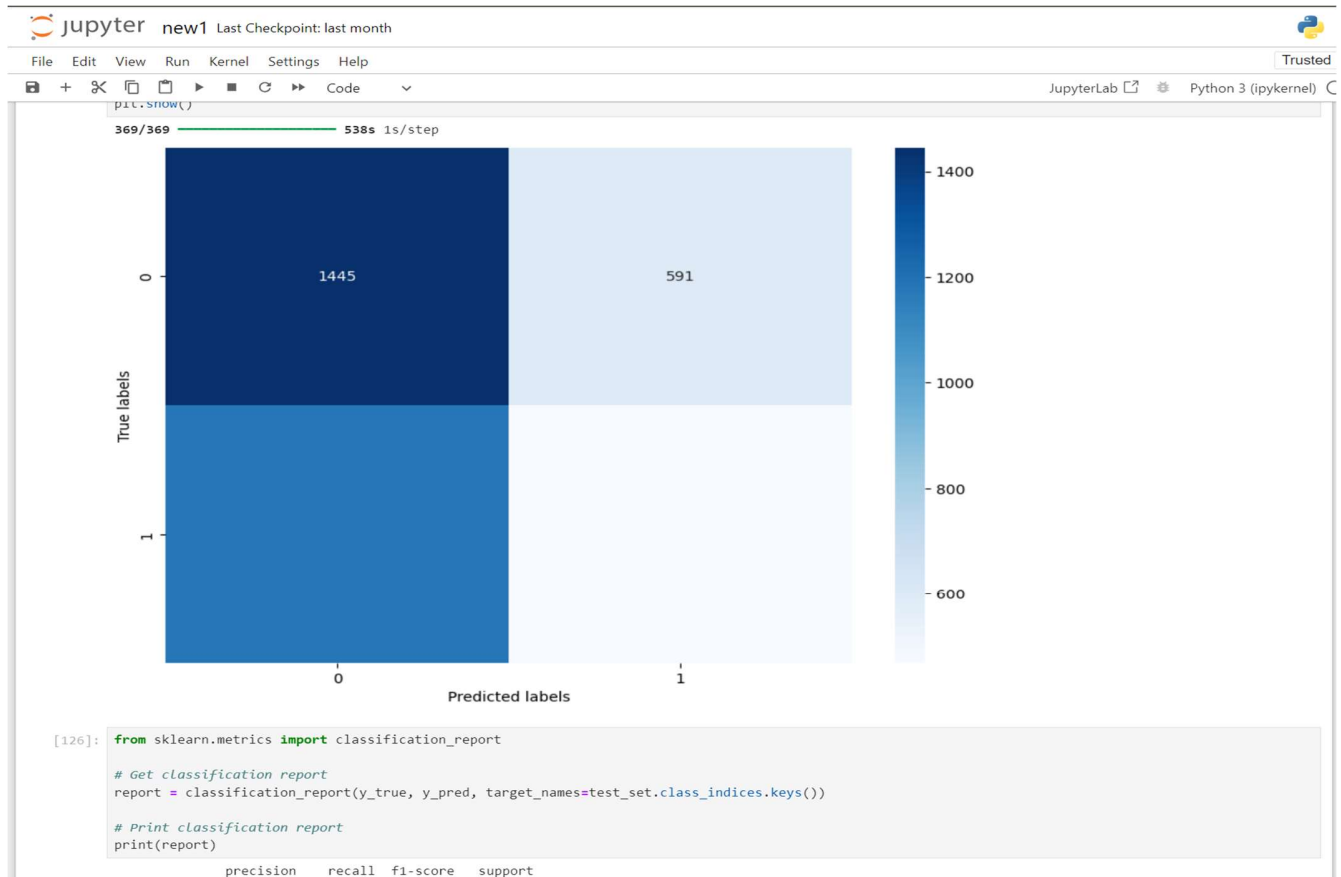
[124]: from sklearn.metrics import confusion_matrix
import seaborn as sns

# Get the true labels
y_true = test_set.classes

# Get the predicted labels
y_pred = model.predict(test_set)
y_pred = np.argmax(y_pred, axis=1)

# Generate confusion matrix
cm = confusion_matrix(y_true, y_pred)

# Plot confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.show()
```



predicted labels

```
[126]: from sklearn.metrics import classification_report

# Get classification report
report = classification_report(y_true, y_pred, target_names=test_set.class_indices.keys())

# Print classification report
print(report)
```

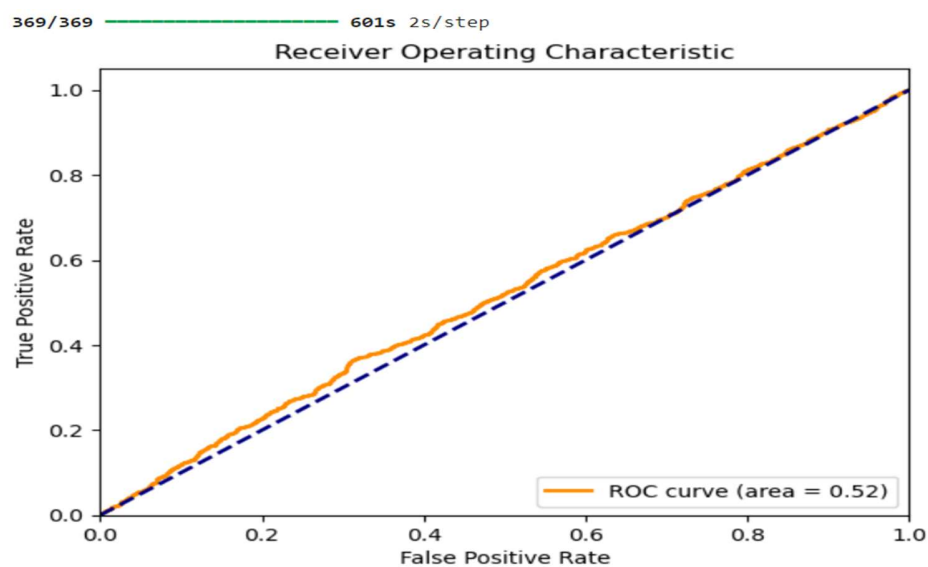
	precision	recall	f1-score	support
Normal	0.55	0.71	0.62	2036
Tuberculosis	0.44	0.28	0.35	1646
accuracy			0.52	3682
macro avg	0.50	0.50	0.48	3682
weighted avg	0.50	0.52	0.50	3682

```
[128]: from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Get predicted probabilities for class 1
y_pred_prob = model.predict(test_set)[: , 1]

# Compute ROC curve and ROC area for each class
fpr, tpr, _ = roc_curve(y_true, y_pred_prob)
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc='lower right')
plt.show()
```



the model.

```
[52]: # Load and preprocess a single image for prediction
img_path = 'Data/ICMR/train/Tuberculosis/Tuberculosis-462.png'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
img_data = preprocess_input(x)

# Predict the class of the image
classes = model.predict(img_data)

# Interpret and print the prediction result
result = np.argmax(classes, axis=1)[0]
if result == 0:
    print("Result is Normal")
else:
    print("Person affected by Tuberculosis")
```

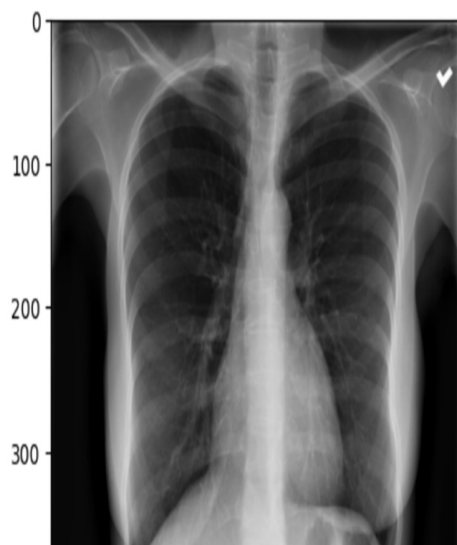
1/1 ————— 0s 120ms/step  
Person affected by Tuberculosis

```
[108]: import cv2
```

```
[110]: test_imgae=cv2.imread(img_path)
```

```
[112]: plt.imshow(test_imgae)
```

```
[112]: <matplotlib.image.AxesImage at 0x1bfcba89490>
```



## Programming for RESNET50 architecture

Jupyter Untitled12 Last Checkpoint: 3 days ago



File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

```
[3]: import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, f1_score, roc_curve, auc, classification_report
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.optimizers import Adam
```

```
[5]: # Load the pre-trained ResNet50 model and fine-tune it
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
x = base_model.output
x = Flatten()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(1, activation='sigmoid')(x) # Binary classification (Normal vs Tuberculosis)

model = Model(inputs=base_model.input, outputs=predictions)
```

```
[7]: # Freeze the layers of the base model
for layer in base_model.layers:
    layer.trainable = False
```

```
[9]: # Compile the model
model.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])
```

```
[11]: # Load and preprocess the dataset
train_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
train_set = train_datagen.flow_from_directory('Data/ICMR/train_', target_size=(224, 224), batch_size=32, class_mode='binary')
test_set = test_datagen.flow_from_directory('Data/ICMR/test_', target_size=(224, 224), batch_size=32, class_mode='binary', shuffle=False)
```

Found 6465 images belonging to 2 classes.  
Found 1275 images belonging to 2 classes.

```
[15]: train_set.class_indices
```

```
[15]: {'Normal': 0, 'Tuberculosis': 1}
```

```
[17]: test_set.class_indices

[17]: {'Normal': 0, 'Tuberculosis': 1}

[19]: # Train the model
model.fit(train_set, epochs=10, validation_data=test_set)

Epoch 1/10
C:\Users\Public\Anaconda\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
203/203 ————— 6518s 32s/step - accuracy: 0.6453 - loss: 21.2615 - val_accuracy: 0.7067 - val_loss: 0.7481
Epoch 2/10
203/203 ————— 1104s 5s/step - accuracy: 0.8699 - loss: 0.3179 - val_accuracy: 0.6839 - val_loss: 0.7567
Epoch 3/10
203/203 ————— 2804s 6s/step - accuracy: 0.9365 - loss: 0.1662 - val_accuracy: 0.7106 - val_loss: 0.8148
Epoch 4/10
203/203 ————— 14487s 72s/step - accuracy: 0.9656 - loss: 0.1034 - val_accuracy: 0.6941 - val_loss: 0.8347
Epoch 5/10
203/203 ————— 1399s 7s/step - accuracy: 0.9899 - loss: 0.0521 - val_accuracy: 0.7263 - val_loss: 0.9040
Epoch 6/10
203/203 ————— 927s 4s/step - accuracy: 0.9943 - loss: 0.0361 - val_accuracy: 0.6784 - val_loss: 1.2668
Epoch 7/10
203/203 ————— 924s 4s/step - accuracy: 0.9944 - loss: 0.0280 - val_accuracy: 0.7043 - val_loss: 1.0046
Epoch 8/10
203/203 ————— 931s 5s/step - accuracy: 1.0000 - loss: 0.0096 - val_accuracy: 0.7043 - val_loss: 1.0746
Epoch 9/10
203/203 ————— 908s 4s/step - accuracy: 1.0000 - loss: 0.0054 - val_accuracy: 0.7082 - val_loss: 1.0961
Epoch 10/10
203/203 ————— 1905s 7s/step - accuracy: 1.0000 - loss: 0.0038 - val_accuracy: 0.6918 - val_loss: 1.1972

[19]: <keras.src.callbacks.history.History at 0x268ca1ec250>

[22]: model.summary()
```

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 224, 224, 3)	0	-
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_layer_1[0]...
conv1_conv (Conv2D)	(None, 112, 112, 64)	9,472	conv1_pad[0][0]
conv1_bn (BatchNormalizatio...	(None, 112, 112, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 112, 112, 64)	0	conv1_bn[0][0]
pool1_pad	(None, 114, 114, 64)	0	conv1_relu[0][0]

(ZeroPadding2D)	64)		
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4,160	pool1_pool[0][0]
conv2_block1_1_bn (BatchNormalizatio...	(None, 56, 56, 64)	256	conv2_block1_1_c...
conv2_block1_1_relu (Activation)	(None, 56, 56, 64)	0	conv2_block1_1_b...
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36,928	conv2_block1_1_r...
conv2_block1_2_bn (BatchNormalizatio...	(None, 56, 56, 64)	256	conv2_block1_2_c...
conv2_block1_2_relu (Activation)	(None, 56, 56, 64)	0	conv2_block1_2_b...
conv2_block1_0_conv (Conv2D)	(None, 56, 56, 256)	16,640	pool1_pool[0][0]
conv2_block1_3_conv (Conv2D)	(None, 56, 56, 256)	16,640	conv2_block1_2_r...
conv2_block1_0_bn (BatchNormalizatio...	(None, 56, 56, 256)	1,024	conv2_block1_0_c...
conv2_block1_3_bn (BatchNormalizatio...	(None, 56, 56, 256)	1,024	conv2_block1_3_c...
conv2_block1_add (Add)	(None, 56, 56, 256)	0	conv2_block1_0_b... conv2_block1_3_b...
conv2_block1_out (Activation)	(None, 56, 56, 256)	0	conv2_block1_add...
conv2_block2_1_conv (Conv2D)	(None, 56, 56, 64)	16,448	conv2_block1_out...
conv2_block2_1_bn	(None, 56, 56,	256	conv2_block2_1_c...

(BatchNormalizatio...	64)		
conv2_block2_1_relu (Activation)	(None, 56, 56, 64)	0	conv2_block2_1_b...
conv2_block2_2_conv (Conv2D)	(None, 56, 56, 64)	36,928	conv2_block2_1_r...
conv2_block2_2_bn (BatchNormalizatio...	(None, 56, 56, 64)	256	conv2_block2_2_c...
conv2_block2_2_relu (Activation)	(None, 56, 56, 64)	0	conv2_block2_2_b...
conv2_block2_3_conv (Conv2D)	(None, 56, 56, 256)	16,640	conv2_block2_2_r...
conv2_block2_3_bn (BatchNormalizatio...	(None, 56, 56, 256)	1,024	conv2_block2_3_c...
conv2_block2_add (Add)	(None, 56, 56, 256)	0	conv2_block1_out... conv2_block2_3_b...
conv2_block2_out (Activation)	(None, 56, 56, 256)	0	conv2_block2_add...
conv2_block3_1_conv (Conv2D)	(None, 56, 56, 64)	16,448	conv2_block2_out...
conv2_block3_1_bn (BatchNormalizatio...	(None, 56, 56, 64)	256	conv2_block3_1_c...
conv2_block3_1_relu (Activation)	(None, 56, 56, 64)	0	conv2_block3_1_b...
conv2_block3_2_conv (Conv2D)	(None, 56, 56, 64)	36,928	conv2_block3_1_r...
conv2_block3_2_bn (BatchNormalizatio...	(None, 56, 56, 64)	256	conv2_block3_2_c...
conv2_block3_2_relu (Activation)	(None, 56, 56, 64)	0	conv2_block3_2_b...
conv2_block3_3_conv	(None, 56, 56,	16,640	conv2_block3_2_r...

(Conv2D)	256)		
conv2_block3_3_bn (BatchNormalizatio...	(None, 56, 56, 256)	1,024	conv2_block3_3_c...
conv2_block3_add (Add)	(None, 56, 56, 256)	0	conv2_block2_out... conv2_block3_3_b...
conv2_block3_out (Activation)	(None, 56, 56, 256)	0	conv2_block3_add...
conv3_block1_1_conv (Conv2D)	(None, 28, 28, 128)	32,896	conv2_block3_out...
conv3_block1_1_bn (BatchNormalizatio...	(None, 28, 28, 128)	512	conv3_block1_1_c...
conv3_block1_1_relu (Activation)	(None, 28, 28, 128)	0	conv3_block1_1_b...
conv3_block1_2_conv (Conv2D)	(None, 28, 28, 128)	147,584	conv3_block1_1_r...
conv3_block1_2_bn (BatchNormalizatio...	(None, 28, 28, 128)	512	conv3_block1_2_c...
conv3_block1_2_relu (Activation)	(None, 28, 28, 128)	0	conv3_block1_2_b...
conv3_block1_0_conv (Conv2D)	(None, 28, 28, 512)	131,584	conv2_block3_out...
conv3_block1_3_conv (Conv2D)	(None, 28, 28, 512)	66,048	conv3_block1_2_r...
conv3_block1_0_bn (BatchNormalizatio...	(None, 28, 28, 512)	2,048	conv3_block1_0_c...
conv3_block1_3_bn (BatchNormalizatio...	(None, 28, 28, 512)	2,048	conv3_block1_3_c...
conv3_block1_add (Add)	(None, 28, 28, 512)	0	conv3_block1_0_b... conv3_block1_3_b...
conv3_block1_out	(None, 28, 28,	0	conv3_block1_add...



(Activation)	512)		
conv3_block2_1_conv (Conv2D)	(None, 28, 28, 128)	65,664	conv3_block1_out...
conv3_block2_1_bn (BatchNormalizatio...	(None, 28, 28, 128)	512	conv3_block2_1_c...
conv3_block2_1_relu (Activation)	(None, 28, 28, 128)	0	conv3_block2_1_b...
conv3_block2_2_conv (Conv2D)	(None, 28, 28, 128)	147,584	conv3_block2_1_r...
conv3_block2_2_bn (BatchNormalizatio...	(None, 28, 28, 128)	512	conv3_block2_2_c...
conv3_block2_2_relu (Activation)	(None, 28, 28, 128)	0	conv3_block2_2_b...
conv3_block2_3_conv (Conv2D)	(None, 28, 28, 512)	66,048	conv3_block2_2_r...
conv3_block2_3_bn (BatchNormalizatio...	(None, 28, 28, 512)	2,048	conv3_block2_3_c...
conv3_block2_add (Add)	(None, 28, 28, 512)	0	conv3_block1_out... conv3_block2_3_b...
conv3_block2_out (Activation)	(None, 28, 28, 512)	0	conv3_block2_add...
conv3_block3_1_conv (Conv2D)	(None, 28, 28, 128)	65,664	conv3_block2_out...
conv3_block3_1_bn (BatchNormalizatio...	(None, 28, 28, 128)	512	conv3_block3_1_c...
conv3_block3_1_relu (Activation)	(None, 28, 28, 128)	0	conv3_block3_1_b...
conv3_block3_2_conv (Conv2D)	(None, 28, 28, 128)	147,584	conv3_block3_1_r...
conv3_block3_2_bn	(None, 28, 28,	512	conv3_block3_2_c...

(BatchNormalizatio...	128)		
conv3_block3_2_relu (Activation)	(None, 28, 28, 128)	0	conv3_block3_2_b...
conv3_block3_3_conv (Conv2D)	(None, 28, 28, 512)	66,048	conv3_block3_2_r...
conv3_block3_3_bn (BatchNormalizatio...	(None, 28, 28, 512)	2,048	conv3_block3_3_c...
conv3_block3_add (Add)	(None, 28, 28, 512)	0	conv3_block2_out... conv3_block3_3_b...
conv3_block3_out (Activation)	(None, 28, 28, 512)	0	conv3_block3_add...
conv3_block4_1_conv (Conv2D)	(None, 28, 28, 128)	65,664	conv3_block3_out...
conv3_block4_1_bn (BatchNormalizatio...	(None, 28, 28, 128)	512	conv3_block4_1_c...
conv3_block4_1_relu (Activation)	(None, 28, 28, 128)	0	conv3_block4_1_b...
conv3_block4_2_conv (Conv2D)	(None, 28, 28, 128)	147,584	conv3_block4_1_r...
conv3_block4_2_bn (BatchNormalizatio...	(None, 28, 28, 128)	512	conv3_block4_2_c...
conv3_block4_2_relu (Activation)	(None, 28, 28, 128)	0	conv3_block4_2_b...
conv3_block4_3_conv (Conv2D)	(None, 28, 28, 512)	66,048	conv3_block4_2_r...
conv3_block4_3_bn (BatchNormalizatio...	(None, 28, 28, 512)	2,048	conv3_block4_3_c...
conv3_block4_add (Add)	(None, 28, 28, 512)	0	conv3_block3_out... conv3_block4_3_b...
conv3_block4_out	(None, 28, 28,	0	conv3_block4_add...

(Activation)	512)		
conv4_block1_1_conv (Conv2D)	(None, 14, 14, 256)	131,328	conv3_block4_out...
conv4_block1_1_bn (BatchNormalizatio...	(None, 14, 14, 256)	1,024	conv4_block1_1_c...
conv4_block1_1_relu (Activation)	(None, 14, 14, 256)	0	conv4_block1_1_b...
conv4_block1_2_conv (Conv2D)	(None, 14, 14, 256)	590,080	conv4_block1_1_r...
conv4_block1_2_bn (BatchNormalizatio...	(None, 14, 14, 256)	1,024	conv4_block1_2_c...
conv4_block1_2_relu (Activation)	(None, 14, 14, 256)	0	conv4_block1_2_b...
conv4_block1_0_conv (Conv2D)	(None, 14, 14, 1024)	525,312	conv3_block4_out...
conv4_block1_3_conv (Conv2D)	(None, 14, 14, 1024)	263,168	conv4_block1_2_r...
conv4_block1_0_bn (BatchNormalizatio...	(None, 14, 14, 1024)	4,096	conv4_block1_0_c...
conv4_block1_3_bn (BatchNormalizatio...	(None, 14, 14, 1024)	4,096	conv4_block1_3_c...
conv4_block1_add (Add)	(None, 14, 14, 1024)	0	conv4_block1_0_b... conv4_block1_3_b...
conv4_block1_out (Activation)	(None, 14, 14, 1024)	0	conv4_block1_add...
conv4_block2_1_conv (Conv2D)	(None, 14, 14, 256)	262,400	conv4_block1_out...
conv4_block2_1_bn (BatchNormalizatio...	(None, 14, 14, 256)	1,024	conv4_block2_1_c...
conv4_block2_1_relu	(None, 14, 14,	0	conv4_block2_1_b...

(Activation)	256)		
conv4_block2_2_conv (Conv2D)	(None, 14, 14, 256)	590,080	conv4_block2_1_r...
conv4_block2_2_bn (BatchNormalizatio...	(None, 14, 14, 256)	1,024	conv4_block2_2_c...
conv4_block2_2_relu (Activation)	(None, 14, 14, 256)	0	conv4_block2_2_b...
conv4_block2_3_conv (Conv2D)	(None, 14, 14, 1024)	263,168	conv4_block2_2_r...
conv4_block2_3_bn (BatchNormalizatio...	(None, 14, 14, 1024)	4,096	conv4_block2_3_c...
conv4_block2_add (Add)	(None, 14, 14, 1024)	0	conv4_block1_out... conv4_block2_3_b...
conv4_block2_out (Activation)	(None, 14, 14, 1024)	0	conv4_block2_add...
conv4_block3_1_conv (Conv2D)	(None, 14, 14, 256)	262,400	conv4_block2_out...
conv4_block3_1_bn (BatchNormalizatio...	(None, 14, 14, 256)	1,024	conv4_block3_1_c...
conv4_block3_1_relu (Activation)	(None, 14, 14, 256)	0	conv4_block3_1_b...
conv4_block3_2_conv (Conv2D)	(None, 14, 14, 256)	590,080	conv4_block3_1_r...
conv4_block3_2_bn (BatchNormalizatio...	(None, 14, 14, 256)	1,024	conv4_block3_2_c...
conv4_block3_2_relu (Activation)	(None, 14, 14, 256)	0	conv4_block3_2_b...
conv4_block3_3_conv (Conv2D)	(None, 14, 14, 1024)	263,168	conv4_block3_2_r...
conv4_block3_3_bn	(None, 14, 14,	4,096	conv4_block3_3_c...

(BatchNormalizatio...	1024)		
conv4_block3_add (Add)	(None, 14, 14, 1024)	0	conv4_block2_out... conv4_block3_3_b...
conv4_block3_out (Activation)	(None, 14, 14, 1024)	0	conv4_block3_add...
conv4_block4_1_conv (Conv2D)	(None, 14, 14, 256)	262,400	conv4_block3_out...
conv4_block4_1_bn (BatchNormalizatio...	(None, 14, 14, 256)	1,024	conv4_block4_1_c...
conv4_block4_1_relu (Activation)	(None, 14, 14, 256)	0	conv4_block4_1_b...
conv4_block4_2_conv (Conv2D)	(None, 14, 14, 256)	590,080	conv4_block4_1_r...
conv4_block4_2_bn (BatchNormalizatio...	(None, 14, 14, 256)	1,024	conv4_block4_2_c...
conv4_block4_2_relu (Activation)	(None, 14, 14, 256)	0	conv4_block4_2_b...
conv4_block4_3_conv (Conv2D)	(None, 14, 14, 1024)	263,168	conv4_block4_2_r...
conv4_block4_3_bn (BatchNormalizatio...	(None, 14, 14, 1024)	4,096	conv4_block4_3_c...
conv4_block4_add (Add)	(None, 14, 14, 1024)	0	conv4_block3_out... conv4_block4_3_b...
conv4_block4_out (Activation)	(None, 14, 14, 1024)	0	conv4_block4_add...
conv4_block5_1_conv (Conv2D)	(None, 14, 14, 256)	262,400	conv4_block4_out...
conv4_block5_1_bn (BatchNormalizatio...	(None, 14, 14, 256)	1,024	conv4_block5_1_c...
conv4_block5_1_relu	(None, 14, 14,	0	conv4_block5_1_b...

(Activation)	256)		
conv4_block5_2_conv (Conv2D)	(None, 14, 14, 256)	590,080	conv4_block5_1_r...
conv4_block5_2_bn (BatchNormalizatio...	(None, 14, 14, 256)	1,024	conv4_block5_2_c...
conv4_block5_2_relu (Activation)	(None, 14, 14, 256)	0	conv4_block5_2_b...
conv4_block5_3_conv (Conv2D)	(None, 14, 14, 1024)	263,168	conv4_block5_2_r...
conv4_block5_3_bn (BatchNormalizatio...	(None, 14, 14, 1024)	4,096	conv4_block5_3_c...
conv4_block5_add (Add)	(None, 14, 14, 1024)	0	conv4_block4_out... conv4_block5_3_b...
conv4_block5_out (Activation)	(None, 14, 14, 1024)	0	conv4_block5_add...
conv4_block6_1_conv (Conv2D)	(None, 14, 14, 256)	262,400	conv4_block5_out...
conv4_block6_1_bn (BatchNormalizatio...	(None, 14, 14, 256)	1,024	conv4_block6_1_c...
conv4_block6_1_relu (Activation)	(None, 14, 14, 256)	0	conv4_block6_1_b...
conv4_block6_2_conv (Conv2D)	(None, 14, 14, 256)	590,080	conv4_block6_1_r...
conv4_block6_2_bn (BatchNormalizatio...	(None, 14, 14, 256)	1,024	conv4_block6_2_c...
conv4_block6_2_relu (Activation)	(None, 14, 14, 256)	0	conv4_block6_2_b...
conv4_block6_3_conv (Conv2D)	(None, 14, 14, 1024)	263,168	conv4_block6_2_r...
conv4_block6_3_bn	(None, 14, 14,	4,096	conv4_block6_3_c...

(BatchNormalizatio...	1024)		
conv4_block6_add (Add)	(None, 14, 14, 1024)	0	conv4_block5_out... conv4_block6_3_b...
conv4_block6_out (Activation)	(None, 14, 14, 1024)	0	conv4_block6_add...
conv5_block1_1_conv (Conv2D)	(None, 7, 7, 512)	524,800	conv4_block6_out...
conv5_block1_1_bn (BatchNormalizatio...	(None, 7, 7, 512)	2,048	conv5_block1_1_c...
conv5_block1_1_relu (Activation)	(None, 7, 7, 512)	0	conv5_block1_1_b...
conv5_block1_2_conv (Conv2D)	(None, 7, 7, 512)	2,359,808	conv5_block1_1_r...
conv5_block1_2_bn (BatchNormalizatio...	(None, 7, 7, 512)	2,048	conv5_block1_2_c...
conv5_block1_2_relu (Activation)	(None, 7, 7, 512)	0	conv5_block1_2_b...
conv5_block1_0_conv (Conv2D)	(None, 7, 7, 2048)	2,099,200	conv4_block6_out...
conv5_block1_3_conv (Conv2D)	(None, 7, 7, 2048)	1,050,624	conv5_block1_2_r...
conv5_block1_0_bn (BatchNormalizatio...	(None, 7, 7, 2048)	8,192	conv5_block1_0_c...
conv5_block1_3_bn (BatchNormalizatio...	(None, 7, 7, 2048)	8,192	conv5_block1_3_c...
conv5_block1_add (Add)	(None, 7, 7, 2048)	0	conv5_block1_0_b... conv5_block1_3_b...
conv5_block1_out (Activation)	(None, 7, 7, 2048)	0	conv5_block1_add...
conv5_block2_1_conv	(None, 7, 7, 512)	1,049,088	conv5_block1_out...

(Conv2D)			
conv5_block2_1_bn (BatchNormalizatio...	(None, 7, 7, 512)	2,048	conv5_block2_1_c...
conv5_block2_1_relu (Activation)	(None, 7, 7, 512)	0	conv5_block2_1_b...
conv5_block2_2_conv (Conv2D)	(None, 7, 7, 512)	2,359,808	conv5_block2_1_r...
conv5_block2_2_bn (BatchNormalizatio...	(None, 7, 7, 512)	2,048	conv5_block2_2_c...
conv5_block2_2_relu (Activation)	(None, 7, 7, 512)	0	conv5_block2_2_b...
conv5_block2_3_conv (Conv2D)	(None, 7, 7, 2048)	1,050,624	conv5_block2_2_r...
conv5_block2_3_bn (BatchNormalizatio...	(None, 7, 7, 2048)	8,192	conv5_block2_3_c...
conv5_block2_add (Add)	(None, 7, 7, 2048)	0	conv5_block1_out... conv5_block2_3_b...
conv5_block2_out (Activation)	(None, 7, 7, 2048)	0	conv5_block2_add...
conv5_block3_1_conv (Conv2D)	(None, 7, 7, 512)	1,049,088	conv5_block2_out...
conv5_block3_1_bn (BatchNormalizatio...	(None, 7, 7, 512)	2,048	conv5_block3_1_c...
conv5_block3_1_relu (Activation)	(None, 7, 7, 512)	0	conv5_block3_1_b...
conv5_block3_2_conv (Conv2D)	(None, 7, 7, 512)	2,359,808	conv5_block3_1_r...
conv5_block3_2_bn (BatchNormalizatio...	(None, 7, 7, 512)	2,048	conv5_block3_2_c...
conv5_block3_2_relu	(None, 7, 7, 512)	0	conv5_block3_2_b...



(Activation)			
conv5_block3_3_conv (Conv2D)	(None, 7, 7, 2048)	1,050,624	conv5_block3_2_r...
conv5_block3_3_bn (BatchNormalizatio...	(None, 7, 7, 2048)	8,192	conv5_block3_3_c...
conv5_block3_add (Add)	(None, 7, 7, 2048)	0	conv5_block2_out... conv5_block3_3_b...
conv5_block3_out (Activation)	(None, 7, 7, 2048)	0	conv5_block3_add...
flatten_1 (Flatten)	(None, 100352)	0	conv5_block3_out...
dense_2 (Dense)	(None, 1024)	102,761,4...	flatten_1[0][0]
dense_3 (Dense)	(None, 1)	1,025	dense_2[0][0]

**Total params:** 331,875,205 (1.24 GB)

**Trainable params:** 102,762,497 (392.01 MB)

**Non-trainable params:** 23,587,712 (89.98 MB)

**Optimizer params:** 205,524,996 (784.02 MB)

```
[25]: # Make predictions on the test set
test_set.reset()
y_pred_prob = model.predict(test_set)
y_pred = (y_pred_prob > 0.5).astype(int)
y_true = test_set.classes
```

40/40 ————— 132s 3s/step

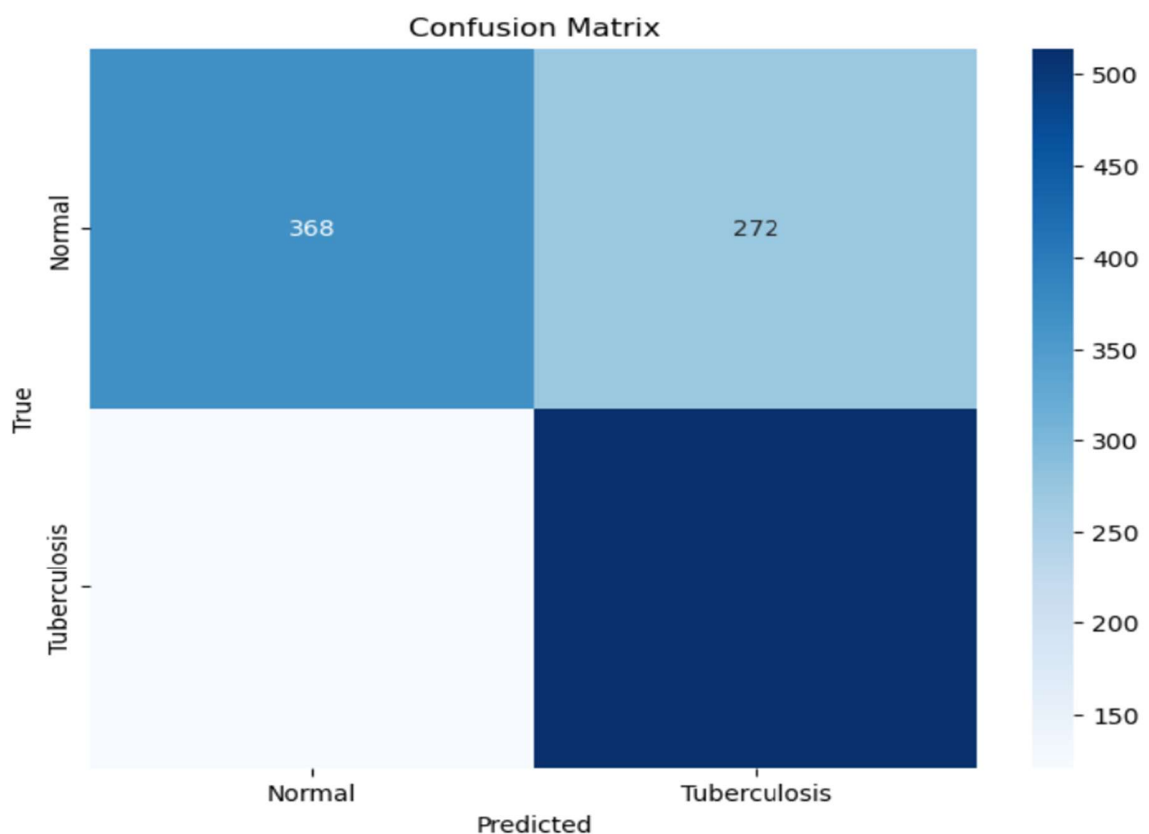
```
[27]: # Generate classification report
class_names = list(test_set.class_indices.keys())
report = classification_report(y_true, y_pred, target_names=class_names)
print(report)
```

	precision	recall	f1-score	support
Normal	0.75	0.57	0.65	640
Tuberculosis	0.65	0.81	0.72	635
accuracy			0.69	1275
macro avg	0.70	0.69	0.69	1275
weighted avg	0.70	0.69	0.69	1275

```
[29]: # Compute confusion matrix
cm = confusion_matrix(y_true, y_pred)

# Plot confusion matrix
def plot_confusion_matrix(cm, class_names):
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title('Confusion Matrix')
    plt.show()

plot_confusion_matrix(cm, class_names)
```



```
[31]: # Calculate F1 score
```

```
f1 = f1_score(y_true, y_pred)
```

```
print(f"F1 Score: {f1:.2f}")
```

F1 Score: 0.72

```
[33]: # Compute ROC curve and ROC area
```

```
fpr, tpr, _ = roc_curve(y_true, y_pred_prob)
```

```
roc_auc = auc(fpr, tpr)
```

```
# Plot ROC curve
```

```
plt.figure(figsize=(8, 6))
```

```
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
```

```
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--')
```

```
plt.xlim([0.0, 1.0])
```

```
plt.ylim([0.0, 1.05])
```

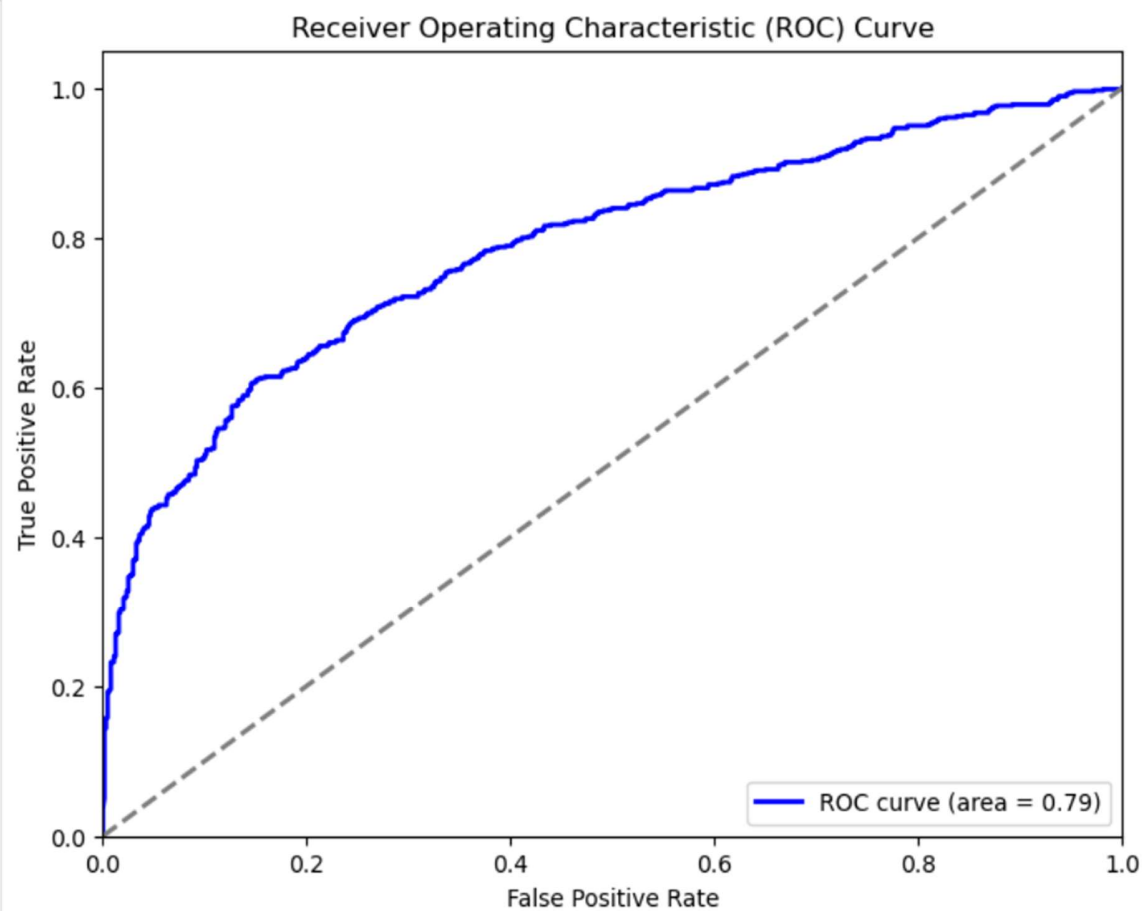
```
plt.xlabel('False Positive Rate')
```

```
plt.ylabel('True Positive Rate')
```

```
plt.title('Receiver Operating Characteristic (ROC) Curve')
```

```
plt.legend(loc='lower right')
```

```
plt.show()
```



NameError: name load\_model is not defined

[61]:

```
# Load and preprocess a single image for prediction
img_path = 'Data/ICMR/train_Normal/normal_02 (3125).jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
img_data = preprocess_input(x)

# Predict the class of the image
classes = model.predict(img_data)

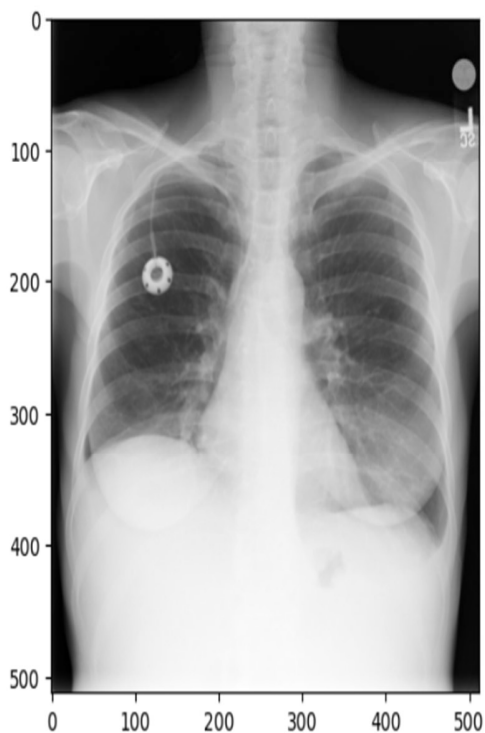
# Interpret and print the prediction result
result = np.argmax(classes, axis=1)[0]
if result == 0:
    print("Result is Normal")
else:
    print("Person affected by Tuberculosis")
```

1/1 ————— 0s 102ms/step  
Result is Normal

[47]:

```
import cv2
test_imgae=cv2.imread(img_path)
plt.imshow(test_imgae)
```

[47]: <matplotlib.image.AxesImage at 0x268d0188e90>



## Programming for the CNN architecture

Jupyter Untitled10 Last Checkpoint: 3 days ago



File Edit View Run Kernel Settings Help

Truste

+



JupyterLab Python 3 (ipykernel)

```
[6]: # Import necessary Libraries
import numpy as np
import matplotlib.pyplot as plt
from keras.layers import Dense, Conv2D, MaxPooling2D, Dropout, Flatten
from keras.models import Sequential
from keras.preprocessing import image
```

```
[12]: from tensorflow.keras.preprocessing import image
```

```
train_datagen = image.ImageDataGenerator(
    rescale=1/255,
    horizontal_flip=True,
    zoom_range=0.2,
    shear_range=0.2
)
```

```
train_data = train_datagen.flow_from_directory(
    directory="Data/ICMR/train_",
    target_size=(256, 256),
    batch_size=1,
    class_mode="binary"
)
```

Found 6465 images belonging to 2 classes.

```
[14]: train_data.class_indices
```

```
[14]: {'Normal': 0, 'Tuberculosis': 1}
```

```
[16]: test_datagen = image.ImageDataGenerator(rescale=1/255)
```

```
test_data = test_datagen.flow_from_directory(
    directory="Data/ICMR/test_",
    target_size=(256, 256),
    batch_size=16,
    class_mode="binary"
)
```

Found 1275 images belonging to 2 classes.

```
[18]: test_data.class_indices
```

```
[18]: {'Normal': 0, 'Tuberculosis': 1}
```

```
[24]: model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation="relu", input_shape=(256, 256, 3)))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation="relu"))
model.add(MaxPooling2D())
model.add(Dropout(rate=0.25))

model.add(Conv2D(filters=64, kernel_size=(3, 3), activation="relu"))
model.add(MaxPooling2D())
model.add(Dropout(rate=0.25))

model.add(Conv2D(filters=128, kernel_size=(3, 3), activation="relu"))
model.add(MaxPooling2D())
model.add(Dropout(rate=0.25))

model.add(Flatten())
model.add(Dense(units=64, activation="relu"))
model.add(Dropout(rate=0.50))
model.add(Dense(units=1, activation="sigmoid"))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
[27]: model.summary()
```

Model: "sequential\_2"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 254, 254, 32)	896
conv2d_7 (Conv2D)	(None, 252, 252, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 126, 126, 64)	0
dropout_4 (Dropout)	(None, 126, 126, 64)	0
conv2d_8 (Conv2D)	(None, 124, 124, 64)	36,928
max_pooling2d_4 (MaxPooling2D)	(None, 62, 62, 64)	0
dropout_5 (Dropout)	(None, 62, 62, 64)	0
conv2d_9 (Conv2D)	(None, 60, 60, 128)	73,856
max_pooling2d_5 (MaxPooling2D)	(None, 30, 30, 128)	0
dropout_6 (Dropout)	(None, 30, 30, 128)	0
flatten_1 (Flatten)	(None, 115200)	0
dense_2 (Dense)	(None, 64)	7,372,864
dropout_7 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 1)	65

Total params: 7,503,105 (28.62 MB)

Trainable params: 7,503,105 (28.62 MB)

Non-trainable params: 0 (0.00 B)

```
[31]: model.fit(
    train_data,
    steps_per_epoch=8,
    epochs=10,
    validation_data=test_data,
    validation_steps=2)
```

Epoch 1/10

C:\Users\Public\Anaconda\Lib\site-packages\keras\src\trainers\data\_adapters\py\_dataset\_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().\_\_init\_\_(\*\*kwargs)` in its constructor. `\*\*kwargs` can include `workers`, `use\_multiprocessing`, `max\_queue\_size`. Do not pass these arguments to `fit()`, as they will be ignored.

self.\_warn\_if\_super\_not\_called()

8/8 ————— 8s 801ms/step - accuracy: 0.4717 - loss: 3.8363 - val\_accuracy: 0.4688 - val\_loss: 0.7164

Epoch 2/10

8/8 ————— 3s 437ms/step - accuracy: 0.3394 - loss: 0.7942 - val\_accuracy: 0.5312 - val\_loss: 0.6928

Epoch 3/10

8/8 ————— 3s 461ms/step - accuracy: 0.3976 - loss: 0.7348 - val\_accuracy: 0.6250 - val\_loss: 0.6924

Epoch 4/10

8/8 ————— 3s 454ms/step - accuracy: 0.5561 - loss: 0.6969 - val\_accuracy: 0.5938 - val\_loss: 0.6925

Epoch 5/10

8/8 ————— 3s 461ms/step - accuracy: 0.1058 - loss: 0.7065 - val\_accuracy: 0.3750 - val\_loss: 0.6939

Epoch 6/10

8/8 ————— 3s 468ms/step - accuracy: 0.3437 - loss: 0.6965 - val\_accuracy: 0.5000 - val\_loss: 0.6932

Epoch 7/10

8/8 ————— 4s 493ms/step - accuracy: 0.4280 - loss: 0.6935 - val\_accuracy: 0.4688 - val\_loss: 0.6932

Epoch 8/10

8/8 ————— 3s 476ms/step - accuracy: 0.2772 - loss: 0.6939 - val\_accuracy: 0.4375 - val\_loss: 0.6932

Epoch 9/10

8/8 ————— 3s 480ms/step - accuracy: 0.1558 - loss: 0.6948 - val\_accuracy: 0.3438 - val\_loss: 0.6935

Epoch 10/10

8/8 ————— 3s 450ms/step - accuracy: 0.6209 - loss: 0.6930 - val\_accuracy: 0.4062 - val\_loss: 0.6936

[31]: <keras.src.callbacks.history.History at 0x16b9cb6f410>



```

# Compute confusion matrix
cm = confusion_matrix(val_labels, val_predictions)

# Plot confusion matrix
def plot_confusion_matrix(cm, class_names):
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title('Confusion Matrix')
    plt.show()

# Calculate F1 score
f1 = f1_score(val_labels, val_predictions)
print(f"F1 Score: {f1:.2f}")

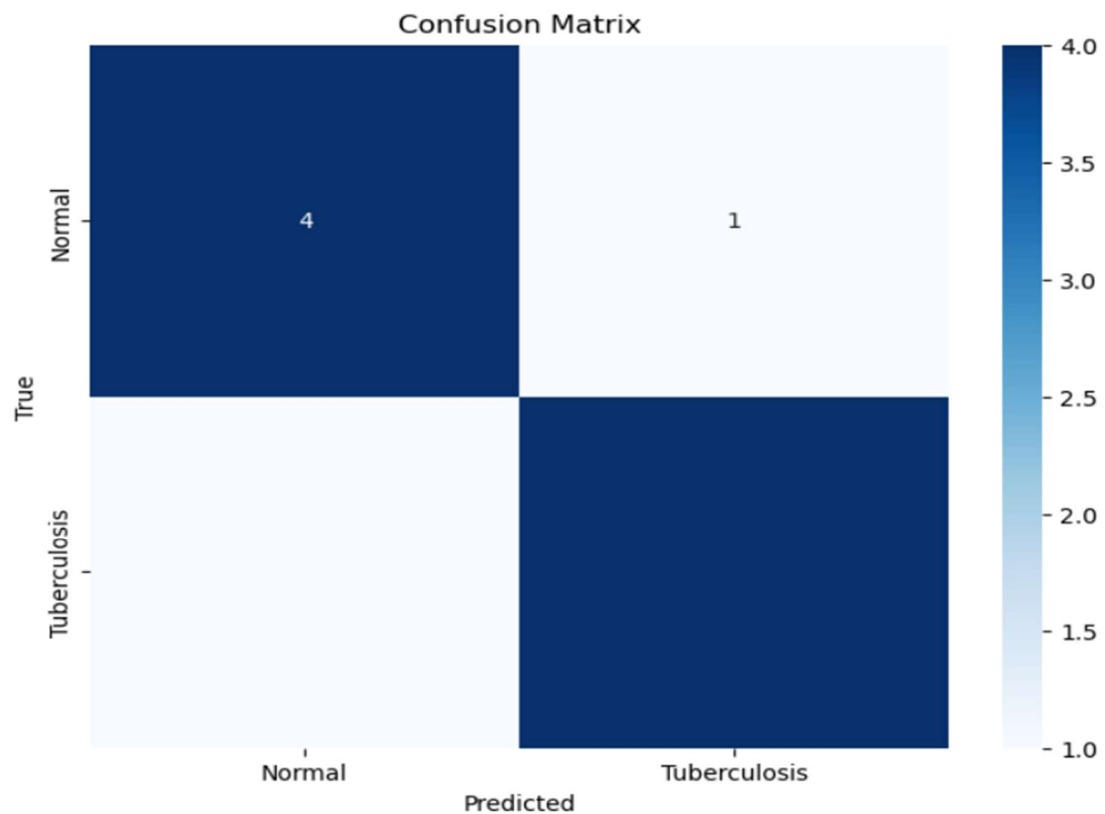
# Plot the confusion matrix
plot_confusion_matrix(cm, class_names=['Normal', 'Tuberculosis'])

# Compute ROC curve and ROC area
fpr, tpr, _ = roc_curve(val_labels, val_predictions_prob)
roc_auc = auc(fpr, tpr)

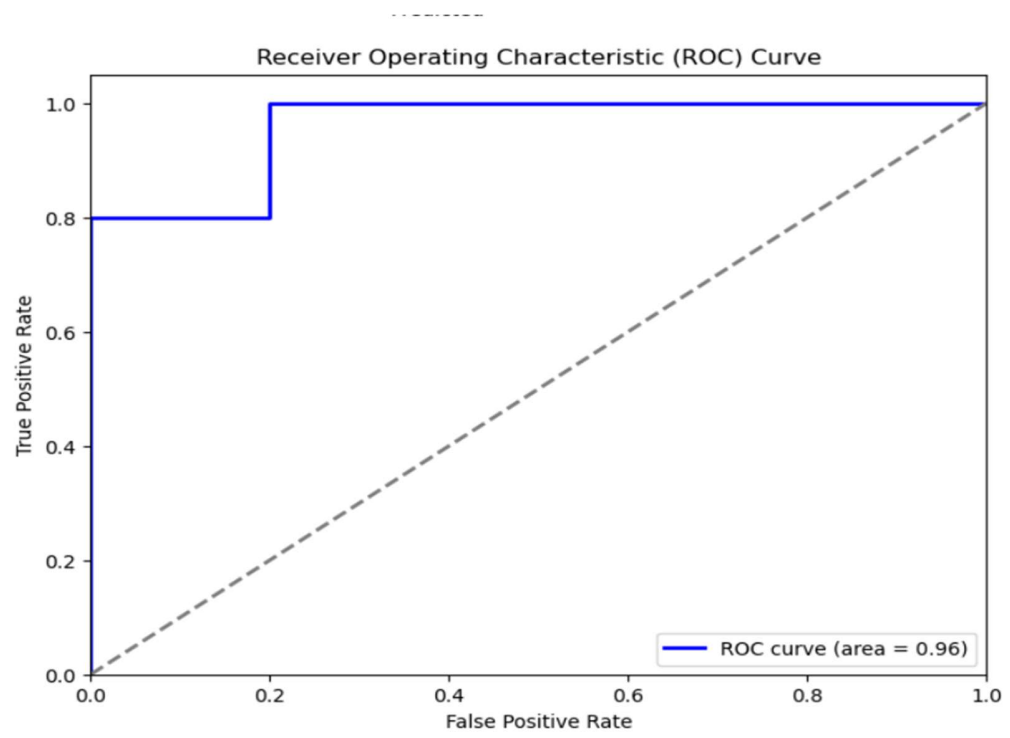
# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()

```

F1 Score: 0.80







```
[101]: # Load and preprocess a single image for prediction
img_path = 'Data/ICMR/train_Normal/normal_02 (3125).jpg'
img = image.load_img(img_path, target_size=(256, 256))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
img_data = preprocess_input(x)

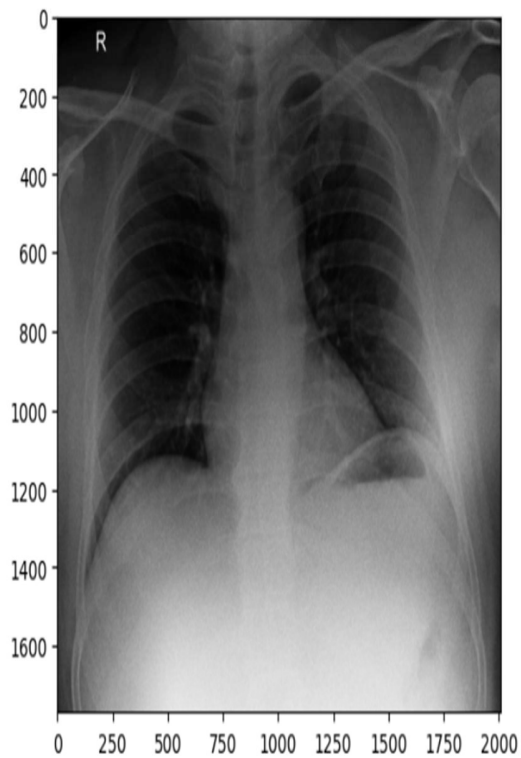
# Predict the class of the image
classes = model.predict(img_data)

# Interpret and print the prediction result
result = np.argmax(classes, axis=1)[0]
if result == 1:
    print("Person is Affected By Tuberculosis")
else:
    print("Result is Normal")
```

1/1 ————— 0s 50ms/step  
Result is Normal

```
[97]: import cv2
test_imgae=cv2.imread(img_path)
plt.imshow(test_imgae)
```

[97]: <matplotlib.image.AxesImage at 0x16be6a19f10>



## Bibliography

1. Showkatian E, Salehi M, Ghaffari H, Reiazi R, Sadighi N. Deep learning-based automatic detection of tuberculosis disease in chest X-ray images. *Polish Journal of Radiology*. 2022;87(1):118–24.
2. Hansun S, Argha A, Liaw ST, Celler BG, Marks GB. Machine and Deep Learning for Tuberculosis Detection on Chest X-Rays: Systematic Literature Review. *Journal of Medical Internet Research* [Internet]. 2023 Jul 3;25:e43154. Available from: <https://pubmed.ncbi.nlm.nih.gov/37399055/>
3. Lakhani, P., & Sundaram, B. (2017). Deep learning at chest radiography: Automated classification of pulmonary tuberculosis by using convolutional neural networks. *Radiology*, 284(2), 574-582.
4. Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A., Ciompi, F., Ghafoorian, M., ... & van Ginneken, B. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, 42, 60-88.
5. Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639), 115-118.
6. Bar, Y., Diamant, I., Wolf, L., & Greenspan, H. (2015). Chest pathology detection using deep learning with non-medical training. In 2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI) (pp. 294-297). IEEE.

## Plagiarism Report

### Ranjeet D report

#### ORIGINALITY REPORT

13%

SIMILARITY INDEX

6%

INTERNET SOURCES

7%

PUBLICATIONS

4%

STUDENT PAPERS

#### PRIMARY SOURCES

1	Eman Showkatian, Mohammad Salehi, Hamed Ghaffari, Reza Reiazi, Nahid Sadighi. "Deep learning-based automatic detection of tuberculosis disease in chest X-ray images", Polish Journal of Radiology, 2022 Publication	4%
2	Submitted to National & Kapodistrian University of Athens Student Paper	2%
3	link.springer.com Internet Source	1%
4	Submitted to Liverpool John Moores University Student Paper	1%
5	Submitted to National Tsing Hua University Student Paper	1%
6	www.geeksforgeeks.org Internet Source	1%
7	Farrukh, Wardah. "Lip Print Based Authentication in Physical Access Control	1%