AN EFFECTIVE APPROACH TO DIAGNOSING TUBERCULOSIS USING DEEP LEARNING TECHNIQUE

CO8811 – PROJECT REPORT

Submitted by

KADIR P	**211420118030**
RAJA VIGNESH P	**211420118037**
SANTHOSH KUMAR P	**211420118042**
SHRIVETHAN P	**211420118046**

in partial fulfillment for the award the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER AND COMMUNICATION ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MARCH 2024

BONAFIDE CERTIFICATE

Certified that this project report **"AN EFFECTIVE APPROACH TO DIAGNOSING TUBERCULOSIS USING DEEP LEARNING TECHNIQUE"** is the Bonafide work of **KADIR P (211420118030)**, **RAJA VIGNESH P (211420118037), SANTHOSH KUMAR P (211420118042) AND SHRIVETHAN P (211420118046)** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. B. ANNI PRINCY M.E., Ph.D.,	Dr. A. DHANALAKSHMI, ME., Ph.D.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSOCIATE PROFESSOR,
COMPUTER AND COMMUNICATION ENGINEERING,	COMPUTER AND COMMUNICATION ENGINEERING,
PANIMALAR ENGINEERING COLLEGE,	PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI,	NAZARATHPETTAI,
POONAMALLEE,	POONAMALLEE,
CHENNAI- 600123.	CHENNAI- 600123.

Certified that the above candidate(s) was/ were examined in the Anna University

Project Viva-Voce Examination held on........................

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr. P. CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Director's Tmt. **C.VIJAYARAJESWARI, Dr. C. SAKTHIKUMAR, M.E., Ph.D**., and **Dr. SARANYA SREE SAKTHIKUMAR B.E., M.B.A., Ph.D**., for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr. K. MANI, M.E., Ph.D.,** for his timely concern and encouragement provided to us throughout the course.

We thank our HOD of Computer and Communication Engineering Department, **Dr. B. ANNI PRINCY, M.E., Ph.D**., Professor, for the support extended throughout the project.

We would like to thank our **s**upervisor **Dr. A. DHANALAKSHMI, M.E., Ph.D.,** Assistant Professor and all the faculty members of the Department of CCE for their advice and suggestions for the successful completion of the project.

KADIR P

RAJA VIGNESH P

SANTHOSH KUMAR P

SHRIVETHAN

ABSTRACT

Tuberculosis (TB) is an airborne infectious disease and a major health threat that is deleterious in most parts of the world. Most of the diagnostic methods are time consuming as well as unreliable and they were all mostly developed in the last century. Chest radiography is used as the most common method for screening TB in a large population. The success of this method depends solely on the experience and interpretation skills of the radiologist. Convolutional neural networks (CNN) is a deep learning strategy that has gained attention and popularity due to its ability to learn midlevel as well as high-level image representations. In this work, several CNN model used as google net   model were used, which classifies the chest radiographs into TB positive and TB negative classes. This technique offers a comparative study on the various deep learning techniques that can process chest x-rays and are capable of TB detection. The performance of the system is measured on a publicly available dataset: Tuberculosis (TB) Chest X-ray Database. The various CNN models, including the GoogleNet model, to classify chest radiographs into TB positive and TB negative categories. By leveraging the Tuberculosis (TB) Chest X-ray Database, the performance of these CNN-based systems is thoroughly evaluated. This comparative analysis sheds light on the effectiveness of different deep learning techniques in processing chest X-rays for TB detection, potentially paving the way for more reliable and accessible diagnostic tools in the fight against tuberculosis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATION

S.NO	ABBREVATION	EXPANSION
1	CXR	Chest X-ray
2	CNN	Convolutional Neural Network
3	MRI	Magnetic Resonance Imaging
4	TB	Tuberculosis
5	RELU	Rectified Linear Unit
6	MOG	Mixture of Gaussian
7	AMD	Advanced Micro Devices
8	GUI	Graphical User Interface
9	API	Application Programming Interface
10	UML	Unified Modelling Language

CHAPTER 1

INTRODUCTION

1.1 DIGITAL IMAGE PROCESSING

The identification of objects in an image would probably start with image processing techniques such as noise removal, followed by (low-level) feature extraction to locate lines, regions and possibly areas with certain textures.

The clever bit is to interpret collections of these shapes as single objects, e.g. cars on a road, boxes on a conveyor belt or cancerous cells on a microscope slide. One reason this is an AI problem is that an object can appear very different when viewed from different angles or under different lighting. Another problem is deciding what features belong to what object and which are background or shadows etc. The human visual system performs these tasks mostly unconsciously but a computer requires skillful programming and lots of processing power to approach human performance. Manipulating data in the form of an image through several possible techniques. An image is usually interpreted as a two-dimensional array of brightness values, and is most familiarly represented by such patterns as those of a photographic print, slide, television screen, or movie screen. An image can be processed optically or digitally with a computer.

To digitally process an image, it is first necessary to reduce the image to a series of numbers that can be manipulated by the computer. Each number representing the brightness value of the image at a particular location is called a picture element, or pixel. A typical digitized image may have 512 × 512 or roughly 250,000 pixels, although much larger images are becoming common.

Once the image has been digitized, there are three basic operations that can be performed on it in the computer. For a point operation, a pixel value in the output image depends on a single pixel value in the input image. For local operations, several neighboring pixels in the input image determine the value of an output image pixel. In a global operation, all of the input image pixels contribute to an output image pixel value.

These operations, taken singly or in combination, are the means by which the image is enhanced, restored, or compressed. An image is enhanced when it is modified so that the information it contains is more clearly evident, but enhancement can also include making the image more visually appealing.

An example is noise smoothing. To smooth a noisy image, median filtering can be applied with a 3 × 3 pixel window. This means that the value of every pixel in the noisy image is recorded, along with the values of its nearest eight neighbors. These nine numbers are then ordered according to size, and the median is selected as the value for the pixel in the new image. As the 3 × 3 window is moved one pixel at a time across the noisy image, the filtered image is formed.

Another example of enhancement is contrast manipulation, where each pixel's value in the new image depends solely on that pixel's value in the old image; in other words, this is a point operation. Contrast manipulation is commonly performed by adjusting the brightness and contrast controls on a television set, or by controlling the exposure and development time in printmaking. Another point operation is that of pseudo coloring a black-and-white image, by assigning arbitrary colors to the gray levels. This technique is popular in thermograph (the imaging of heat), where hotter objects (with high pixel values) are assigned one color (for example, red), and cool objects (with low pixel values) are assigned another color (for example, blue), with other colors assigned to intermediate values.

Recognizing object classes in real-world images is a standing goal in Computer vision. Conceptually, this is challenging due to large appearance variations of object instances belonging to the same class. Additionally, distortions from background clutter, scale, and viewpoint variations can render appearances of even the same object instance to be vastly different. Further challenges arise from interclass similarity in which instances from different classes can appear very similar. Consequently, models for object classes must be flexible enough to accommodate class variability, yet discriminative enough to sieve out true object instances in cluttered images. These seemingly paradoxical requirements of an object class model make recognition difficult. This paper addresses two goals of recognition are image classification and object detection. The task of image classification is to determine if an object class is present in an image, while object detection localizes all instances of that class from an image. Toward these goals, the main contribution in this paper is an approach for object class recognition that employs edge information only. The novelty of our approach is that we represent contours by very simple and generic shape primitives of line segments and ellipses, coupled with a flexible method to learn discriminative primitive combinations. These primitives are complementary in nature, where line segment models straight contour and ellipse models curved contour. We choose an ellipse as it is one of the simplest circular shapes, yet is sufficiently flexible to model curved shapes. These shape primitives possess several attractive properties. First, unlike edge-based descriptors they support abstract and perceptually meaningful reasoning like parallelism and adjacency. Also, unlike contour fragment features, storage demands by these primitives are independent of object size and are efficiently represented with four parameters for a line and five parameters for an ellipse.

Additionally, matching between primitives can be efficiently computed (e.g., with geometric properties), unlike contour fragments, which require comparisons between individual edge pixels. Finally, as geometric properties are easily scale normalized, they simplify matching across scales. In contrast, contour fragments are not scale invariant, and one is forced either to rescale fragments, which introduces aliasing effects (e.g., when edge pixels are pulled apart), or to resize an image before extracting fragments, which degrades image resolution.

In recent studies it is shown that the generic nature of line segments and ellipses affords them an innate ability to represent complex shapes and structures. While individually less distinctive, by combining a number of these primitives, we empower a combination to be sufficiently discriminative. Here, each combination is a two-layer abstraction of primitives: pairs of primitives (termed shape tokens) at the first layer, and a learned number of shape tokens at the second layer. We do not constrain a combination to have a fixed number of shape-tokens, but allow it to automatically and flexibly adapt to an object class. This number influences a combination's ability to represent shapes, where simple shapes favor fewer shape-tokens than complex ones. Consequently, discriminative combinations of varying complexity can be exploited to represent an object class. We learn this combination by exploiting distinguishing shape, geometric, and structural constraints of an object class. Shape constraints describe the visual aspect of shape tokens, while geometric constraints describe its spatial layout (configurations). Structural constraints enforce possible poses/structures of an object by the relationships (e.g., XOR relationship) between shape-tokens.

1.2 CLASSIFICATION OF IMAGES

There are 3 types of images used in Digital Image Processing. They are

1. Binary Image
2. Gray Scale Image
3. Color Image

1.2.1 BINARY IMAGE

A binary image is a digital image that has only two possible values for each pixel. Typically, the two colors used for a binary image are black and white though any two colors can be used. The color used for the object(s) in the image is the foreground color while the rest of the image is the background color.

Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit (0 or 1).This name black and white, monochrome or monochromatic are often used for this concept, but may also designate any images that have only one sample per pixel, such as grayscale images

Binary images often arise in digital image processing as masks or as the result of certain operations such as segmentation, thresholding, and dithering. Some input/output devices, such as laser printers, fax machines, and bi-level computer displays, can only handle bi-level images.

1.2.2 GRAY SCALE IMAGE

A grayscale Image is digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information. Images of this sort, also known as black-and-white, are composed exclusively of shades of gray(0-255), varying from black (0) at the weakest intensity to white (255) at the strongest.

Grayscale images are distinct from one-bit black-and-white images, which in the context of computer imaging are images with only the two colors, black, and white (also called bi-level or binary images). Grayscale images have many shades of gray in between. Grayscale images are also called monochromatic, denoting the absence of any chromatic variation.

Grayscale images are often the result of measuring the intensity of light at each pixel in a single band of the electromagnetic spectrum (e.g. infrared, visible light, ultraviolet, etc.), and in such cases they are monochromatic proper when only a given frequency is captured. But also they can be synthesized from a full color image; see the section about converting to grayscale.

1.2.3 COLOUR IMAGE

A (digital) color image is a digital image that includes color information for each pixel. Each pixel has a particular value which determines it's appearing color. This value is qualified by three numbers giving the decomposition of the color in the three primary colors Red, Green and Blue. Any color visible to human eye can be represented this way. The decomposition of a color in the three primary colors is quantified by a number between 0 and 255. For example, white will be coded as R = 255, G = 255, B = 255; black will be known as (R,G,B) = (0,0,0); and say, bright pink will be : (255,0,255). In other words, an image is an enormous two-dimensional array of color values, pixels, each of them coded on 3 bytes, representing the three primary colors. This allows the image to contain a total of 256x256x256 = 16.8 million different colors. This technique is also known as RGB encoding, and is specifically adapted to human vision.

Moreover, there are a number of applications where emotion recognition can play an important role including biometric authentication, high-technology surveillance and security systems, image retrieval, and passive demographical data collections.

It is unarguable that face is one the most important feature that characterises human beings. By only looking ones' faces, we are not only able to tell who they are but also perceive a lot of information such as their emotions, ages and genders. This is why emotion recognition by face has received much interest in computer vision research community over past two decades.

1.3 DESCRIPTION

Back propagation

Back propagation is a supervised learning algorithm, for training Multi-layer Perceptions (Artificial Neural Networks).

Background Subtraction Algorithm

An innovative system for detecting and extracting vehicles in traffic surveillance scenes is presented. This system involves locating moving objects present in complex road scenes by implementing an advanced background subtraction methodology. The innovation concerns a histogram-based filtering procedure, which collects scatter background information carried in a series of frames, at pixel level, generating reliable instances of the actual background. The proposed algorithm reconstructs a background instance on demand under any traffic conditions. The background reconstruction algorithm demonstrated a rather robust performance in various operating conditions including unstable lighting, different view-angles and congestion.

Contour Extraction Algorithm

The contour is a curve that connects all points surrounding specific part of an image that has the same color or intensity. To identify hand gestures, the contours of all objects that exist in the threshold image are detected . Then, the biggest contour, which has the biggest calculated area representing the hand’s area, is selected. The Suzuki’s algorithm is used to find the hand contour. The filtered contour points are approximated to another contour that has fewer numbers of points. The contour approximation is the process of finding key vertex points and is used to speed up the calculations and then it consumes low memory size.

Template Matching Algorithm

The template matching algorithm called 1$ algorithm, is mainly used in hand writing recognition by comparing the stored template with the user input. The 1$ algorithm supports many features such as scaling independent feature, rotation independent feature, requiring simple mathematical equations, achieving high detection rate, allowing user to train any number of gestures, and low resources consumption. All of these features make the 1$ a good choice for hand’s contour extraction.

In this paper, the template matching algorithm, 1$ algorithm, is modified to satisfy the real-time constraints to recognize hand gesture. This is achieved by comparing the saved contour templates and the biggest extracted contour from the current frame. The best matching template detected is representing the desired contour of the current frame.

Pre-processing

This step consists of a collection of simple image processing tasks that change the raw input video into a format that can be processed by the next steps. In the early stage of processing, simple temporal or spatial smoothing are used to reduce noise such as rain and snow. In pre-processing, the data format used by the background subtraction algorithm is a very important key. Most of the algorithms handle luminance intensity, which is one scalar value per each pixel. However, color image, in either RGB or HSV colour space, is becoming more popular in the background subtraction literature [8, 15]. These papers argue that colour is better than luminance at identifying objects in low-contrast areas and suppressing shadow cast by moving objects. In addition to colour, pixel-based image features such as spatial and temporal derivatives are sometimes used to incorporate edges and motion information. For example, intensity values and spatial derivatives can be combined to form a single state space for background tracking with the Kalman filter. Plessy et al. combine both spatial and temporal derivatives to form a constant velocity background model for detecting speeding vehicles. The main drawback of adding colour or derived features in background modelling is the extra 130 Computer Science & Information Technology (CS & IT) complexity for model parameter estimation. The increase in complexity is often significant as most background modelling techniques maintain an independent model for each pixel.

Background Modelling

Background modelling is at the heart of any background subtraction algorithm. Much research has been devoted to developing a background model that is robust against environmental changes in the background, but sensitive enough to identify all moving objects of interest. We classify background modelling techniques into two broad categories, no recursive and recursive. They are described in the following subsections. Non-recursive techniques: A non-recursive technique uses a sliding-window approach for background estimation. It stores a buffer of the previous L video frames, and estimates the background image based on the temporal variation of each pixel within the buffer . Recursive Techniques: Recursive techniques do not maintain a buffer for background estimation. Instead, they recursively update a single background model based on each input frame. As a result, input frames from distant past could have an effect on the current background model. Recursive techniques require less storage not like no recursive techniques, but any mistake or error in the background model can lead us to a much longer period of time.

Foreground detection

This step makes us able to compare the input video frame with the background model, and identifies foreground pixels from the input frame. There is some techniques that doesn't use the same image as a background model like the Mixture of Gaussian model (MoG), but the most of techniques use a single image as their background models, so the approach for foreground detection is to check whether the input pixel is different from the corresponding background estimate : $|It(x,y) - Bt(x,y)| > T$ (6) Another popular foreground detection scheme is to threshold based on the normalized statistics: $(| It (x,y) - Bt (x,y)- \mu d \ | / \partial d) > Ts$ (7) Where μd and ∂d are the mean and the standard deviation of $It (x, y) - Bt (x, y)$ for all special locations $(x; y)$. Most schemes determine the foreground threshold T or Ts experimentally . Data Validation. To improve the candidate foreground mask based on information obtained from outside the background model, we define data validation process. All the background models have three main limitations:

First, they ignore any correlation between neighboring pixels; second, the rate of adaptation may not match the moving speed of the foreground objects; and third, non-stationary pixels from moving leaves or shadow cast by moving objects are easily mistaken as true foreground objects. When the background model adapts at a slower rate than the foreground scene, large areas of false foreground, commonly known as ghosts, often occur. Sophisticated vision techniques can also be used to validate foreground detection. Computing optical flow for candidate foreground regions caneliminate ghost objects as they have no motion. Colour segmentation can beused to grow foreground regions by assuming similar colour composition throughout the entire object.

W4 System Method

The next method is based on method used in W4 system. As mentioned in [1-2] and[7], we can say, during training sequence maximum intensity, minimum intensity and maximum intensity difference (Dmax) in successive frame of the pixel is calculated and then foreground pixel is calculated based on following equation.

EUCLIDEAN DISTANCE

Having been fiddling around with distance measures for some time – especially with regard to profile comparison methodologies, I thought it was time I provided a brief and simple overview of Euclidean Distance – and why so many programs give so many completely different estimates of it. This is not because the concept itself changes (that of linear distance), but is due to the way programs/ investigators either transform the data prior to computing the difference, normalize constituent distances via a constant, or re-scale and also geometry for measuring distances between points on a plane, providing a precise to determine lengths and relationships between geometric shapes.

The Euclidean metric (and distance magnitude) is that which corresponds to everyday experience and perceptions. That is, the kind of 1, 2, and 3-Dimensional linear metric world where the distance between any two pointsin space corresponds to the length of a straight line drawn between them. Figure 1 shows the scores of three individuals on two variables

The straight line between each “Person” is the Euclidean distance. There would this be three such distances to compute, one for each person-to-person distance. However, we could also calculate the Euclidean distance between the two variables, given the three person scores on each.

The formula for calculating the distance between the two variables, given three persons scoring on each.

$$d = \sqrt{\sum_{i=1}^{p} (v_{1i} - v_{2i})^2}$$

where the difference between two variables’ values is taken, and squared, and summed for p persons (in our example p=3). Only one distance would be computed – between v1 and v2. Let’s do the calculations for finding the Euclidean distances between the three persons, given their scores on two variables. Euclidean distance is a measure of the straight-line distance between two points in a multi-dimensional space. Specifically, it is utilized in feature extraction to quantify the similarity or dissimilarity between different data points or feature vectors. During classification tasks, it serves as a metric to determine the proximity of a given data point to other points in the feature space, aiding in the identification of patterns and decision-making processes. Moreover, in model evaluation, Euclidean distance can be employed to assess the performance and accuracy of trained models by comparing.

NORMALISED EUCLIDEAN DISTANCE

The problem with the raw distance coefficient is that it has no obvious bound value for the maximum distance, merely one that says 0 = absolute identity. Its range of values vary from 0 (absolute identity) to some maximum possible discrepancy value which remains unknown until specifically computed. Raw Euclidean distance varies as a function of the magnitudes of the observations. Basically, you don't know from its size whether a coefficient indicates a small or large distance. If I divided every person's score by 10 in Table 1, and recomputed the euclidean distance between the persons, I would now obtain distance values of 3.736 for person 1 compared to 2, instead of 37.36. Likewise, 8.06 for person 1 and 3, and 6.01 for persons 2 and 3. The raw distance conveys little information about absolute dissimilarity. So, raw euclidean distance is acceptable only if relative ordering amongst a fixed set of profile attributes is required. But, even here, what does a figure of 37.36 actually convey. If the maximum possible observable distance is 38, then we know that the persons being compared are about as different as they can be. But, if the maximum observable distance is 1000, then suddenly a value of 37.36 seems to indicate a pretty good degree of agreement between two persons. The fact of the matter is that unless we know the maximum possible values for a euclidean distance, we can do little more than rank dissimilarities, without ever knowing whether any or them are actually similar or not to one another in any absolute sense.

A further problem is that raw Euclidean distance is sensitive to the scaling of each constituent variable. For example, comparing persons across variables whose score ranges are dramatically different. Likewise, when developing a matrix of Euclidean coefficients by comparing multiple variables to one another, and where those variables' magnitude ranges are quite different.

CHAPTER 2

LITERATURE SURVEY

Title 1: An Optimal Way for Tuberculosis Detection

Author: Ajmal shan C. K.1, Binoy D. L.2

Year: 2020

Description:

Tuberculosis (TB) is a main global health threat. An estimated one-third of the world's population has been exposed to TB, and millions of new infections are occurring every year. Tuberculosis naturally affects the lungs it also affects the other parts of our body. The advent of new powerful hardware and software techniques has triggered attempts to develop computer-aided diagnostic systems for TB detection in support of inexpensive mass screening in developing countries. In this paper the medical background of TB detection in conventional posterior anterior chest X-rays has been described. In the first step the chest x- rays has been given as an input. In the second step, the selected images are segmented using graph cut segmentation method. In the last step asset of features has been extracted and calculated. Lastly, the multi-support vector machine is applied to classify the extracted feature vectors as normal or abnormal lungs. If it is abnormal, provide the name of the most matching TB manifestation of both lungs.

Title 2: Classification of Lungs Diseases Using Machine Learning Technique

Author: Meet Diwan1, Bhargav Patel2, Jaykumar Shah3

Year: 2021

Description:

The application of contemporary technologies is important to medical progress. To create accurate and specialised treatment choices for a range of ailments, extensive study performed in partnership with researchers, health care professionals, and patients is important. This study aims to identify the degree of accuracy that is acceptable in the medical sector by using deep learning on publicly available data. First, we extracted spectrogram features and labels from the annotated lung sound recordings to feed into our 2D Convolutional Neural Network (CNN) model. In this paper, we solve the problem of medical data scarcity by identifying pulmonary diseases from chest X-Ray pictures using small volume datasets with less than a thousand samples. Several studies have been conducted on the application of deep learning to identify lung disease have been published in the literature. The research goes into the history of deep learning and its applications in pulmonary imaging.

Title 3: Tuberculosis diagnosis using Deep Learning
Author: Lokeshwaran V B1, Monish Kumar R2, Lakshman Raaj S3
Year: 2021

Description:

Tuberculosis (TB) is an airborne infectious disease and a major health threat that is deleterious in most parts of the world. Most of the content diagnostic methods are time consuming as well as unreliable and they were all mostly developed in the last century. Chest radiography is used as the most common method for screening TB in a large population. The success of this method depends solely on the experience and interpretation skills of the radiologist. Convolutional neural networks (CNN) is a deep learning strategy that has gained attention and popularity due to its ability to learn midlevel as well as high-level image representations. In this work, several CNN models such as alexnet were used, which classifies the chest radiographs into TB positive and TB negative classes. This paper offers a comparative study on the various deep learning techniques that can process chest x-rays and are capable of TB detection. The performance of the system is measured on a publicly available dataset: Tuberculosis (TB) Chest X-ray Database. The proposed CNN models trained for TB detection achieve accuracy of more than 80%.

Title 4: CXR Tuberculosis Detection Using MATLAB Image Processing
Author: Mr. P. A. Kamble1, Mr. V. V. Anagire2, Mr. S. N. Chamtagoudar3
Year: 2021

Description:

Tuberculosis (TB) is very dangerous and rapidly spread disease in the world. When left undiagnosed and thus untreated, mortality rates of patients with tuberculosis are high. Standard diagnostics still rely on methods developed in the last century. They are slow and often unreliable. In the investigating cases for suspected tuberculosis (TB), chest radiography is not only the key techniques of diagnosis based on the medical imaging but also the diagnostic radiology. So, Computer aided diagnosis (CAD) has been popular and many researchers are interested in this research areas and different approaches have been proposed for the TB detection and lung decease classification. In this paper we present method for detection of Tuberculosis in CXR image by using

MATLAB which includes Pre-processing of Image, Segmentation and Feature extraction from that image.

Title 5: Segmentation of PET Images for Computer-Aided Functional Quantification of Tuberculosis in Small Animal Models

Author: Brent Foster., Ulas Bagci; Ziyue Xu., Bappaditya Dey., Brian Luna., William Bishai., Sanjay Jain., Daniel J. Mollura

Year: 2020

Description:

Pulmonary infections often cause spatially diffuse and multi-focal radiotracer uptake in positron emission tomography (PET) images, which makes accurate quantification of the disease extent challenging. Image segmentation plays a vital role in quantifying uptake due to the distributed nature of immuno-pathology and associated metabolic activities in pulmonary infection, specifically tuberculosis (TB). In this study, we propose a method to select optimal thresholding levels by utilizing a novel intensity affinity metric within the affinity propagation clustering framework. We tested the proposed method against 70 longitudinal PET images of rabbits infected with TB. The overall dice similarity coefficient between the segmentation from the proposed method and two expert segmentations was found to be 91.25 ±8.01% with a sensitivity of 88.80 ±12.59% and a specificity of 96.01 ±9.20%. High accuracy and heightened efficiency of our proposed method, as compared to other PET image segmentation methods, were reported with various quantification metrics.

Title 6: Deep-learning: A potential method for tuberculosis detection using chest radiography

Author: Sanjeev Sofat, Simranpreet Kaur, Ajay Mittal, Fabrice Meriaudeau

Year: 2019

Description:

Tuberculosis (TB) is a major health threat in the developing countries. Many patients die every year due to lack of treatment and error in diagnosis. Developing a computer-aided diagnosis (CAD) system for TB detection can help in early diagnosis and containing the disease. Most of the current CAD systems use handcrafted features, however, lately there is a shift towards deep-learning-based automatic feature extractors. In this paper, we present a potential method for tuberculosis detection using deep-learning which classifies CXR images into two categories, that is, normal and abnormal. We have used CNN architecture with 7 convolutional layers and 3 fully connected layers. The performance of three different optimizers has been compared. Out of these, Adam optimizer with an overall accuracy of 94.73% and validation accuracy of 82.09% performed best amongst them. All the results are obtained using Montgomery and Shenzhen datasets which are available in public domain.

Title 7: Genome-Wide Analysis of MDR and XDR Tuberculosis from Belarus: Machine-Learning Approach

Author: Roman Sergeevich Sergeev., Ivan S.Kavaliou.,Uladzislau. Sataneuski.,Andrei Gabrielian.,Alex Rosenthal

Year: 2019

Description:

Emergence of drug-resistant microorganisms has been recognized as a serious threat to public health worldwide. This problem is extensively discussed in the context of tuberculosis treatment. Alterations in pathogen genomes are among the main mechanisms by which microorganisms exhibit drug resistance. Analysis of 144 M. tuberculosis strains of different phenotypes including drug susceptible, MDR, and XDR isolated in Belarus was fulfilled in this paper. A wide range of machine learning methods that can discover SNPs related to drug-resistance in the whole bacteria genomes was investigated. Besides single-SNP testing approaches, methods that allow detecting joint effects from interacting SNPs were considered. accuracy=83.73%, AUC=90.71%, precision=84.73%, recall=88.83%, F1 score =89.82%. We proposed a framework for automated selection of the best performing statistical model in terms of recall, precision, and accuracy to identify drug resistance-associated mutations. Analysis of whole-genome sequences often leads to situations where the number of treated features exceeds the number of available observations. For this reason, special attention is paid to fair evaluation of the model prediction quality and minimizing the risk of overfitting while estimating the underlying parameters. Results of our experiments aimed at identifying top-scoring resistance mutations to the major first-line and second-line anti-TB drugs are presented.

Title 8: Point-of-Care 3-D Printed Spectrophotometer for Therapeutic Drug Monitoring in Tuberculosis Patients

Author: Pragya Hooda., Muhammad A. Sami., Umer Hassan

Year: 2021

Description:

Tuberculosis (TB) is a highly infectious disease and remains as one the leading causes of mortality around the globe. Many drugs, such as rifampin, isoniazid, ethambutol, and pyrazinamide, are routinely prescribed for its treatment. The dosage requirements for treatment depend on a patients age, gender, and preexisting conditions, if any. Using therapeutic drug monitoring (TDM), clinicians can observe drug absorbance in a patient and adjust the drug dosage accordingly. Here, we have developed a 3-D printed hand-held portable spectrophotometer, which can be used for quantifying antibiotics in biological samples (e.g., urine) for TDM. Using the portable spectrophotometer, we quantified different amounts of rifampin in phosphate buffer saline and synthetic urine and found R 2 values of 0.92 and 0.94, respectively, compared to the standard clinical instrument. The presented system can potentially aide clinicians in making more informed decisions and serves as a first step toward the development of a fully automated point-of-care (POC) system for TDM in TB patients undergoing therapy.

Title 9: Convolution Neural Network With Coordinate Attention for the Automatic Detection of Pulmonary Tuberculosis Images on Chest X-Rays
Author: Tianhao Xu., Zhenming Yuan
Year: 2022

Description:

Tuberculosis is a chronic respiratory infectious disease that seriously endangers human health. Diagnosis of pulmonary tuberculosis usually depend on the analysis of chest X-rays by radiologists. However, there is a certain misdiagnosis rate with time consuming. Therefore, the purpose of this study is to propose a low-cost and automatic detection method of pulmonary tuberculosis images on chest X-rays to help primary radiologists. A pulmonary tuberculosis classification algorithm based on convolution neural network is proposed, which uses deep learning to classify chest X-ray images. Our method introduces coordinate attention mechanism into convolutional neural network (VGG16), so that the algorithm can capture not only cross-channel information, but also direction sensing and position sensing information, in order to better identify and classify pulmonary tuberculosis images. During the training process, we use the method of transfer learning and freeze network to make the model fit faster. The performance of our method is evaluated on the public dataset of tuberculosis classification of Shenzhen Third Hospital, China. We take the average data through 5-fold cross validation: accuracy=92.73%, AUC=97.71%, precision=92.73%, recall=92.83%, F1 score =92.82%. Compared with the existing end-to-end method based on convolutional neural network (CNN), our method is superior to ConvNet, FPN + Faster RCNN and other methods. The comparison results with other methods show that our method has better accuracy, which can help radiologists make auxiliary diagnosis.

Title 10: A Novel Approach for the Detection of Tuberculosis and Pneumonia Using Chest X-Ray Images for Smart Healthcare Applications

Author: Subrat Kumar Kabi., Rajesh Kumar Tripathy., Dipti Patra., Ganapati Panda

Year: 2023

Description:

The early discrimination of pneumonia and tuberculosis is difficult for radiologists due to similar pathological symptoms in the chest X-ray images. Therefore, there is a requirement for automated methods to detect and classify these two diseases accurately. This letter proposes the multiscale eigen domain gradient boosting (MEGB)-based approach to detect pneumonia and tuberculosis from chest X-ray images. The discrete wavelet transform is employed to evaluate sub bands of chest X-ray images at different decomposition levels. The singular value decomposition (SVD) is utilized in each sub band to evaluate singular values, left eigenmatrix, and right eigenmatrix, respectively. The maximum value of each column of both left and right eigenmatrices and singular values for each sub band of the X-ray image are used as features. All sub band eigen domain feature vectors are concatenated and given to the light gradient boosting model to detect pneumonia and tuberculosis diseases. The performance of the proposed MEGB approach-based detection is evaluated using chest X-ray images from a publicly available database. The suggested MEGB approach has achieved an accuracy value of 96.42%. The suggested approach performs better than the transfer learning and other reported methods to detect pneumonia and tuberculosis using chest X-ray images.

2.1 EXISTING SYSTEM

Traditional methods for tuberculosis detection often rely on manual interpretation of chest X-rays by trained radiologists, leading to time-consuming processes and potential diagnostic errors. Existing automated systems face challenges in achieving high accuracy due to the complexity of image interpretation, highlighting the need for advanced techniques such as deep learning to improve efficiency and reliability. In our research, we propose a model that adds a coordinate attention mechanism module to the convolutional neural network VGG16. This addition enables our method to better focus on location and direction information in tuberculosis images, resulting in improved classification accuracy. Additionally, we utilize the method of freezing the network to expedite the model training process and further enhance network performance. Unlike existing end-to-end methods, our approach does not require ensemble learning for multi-model fusion or extensive computing resources. Furthermore, it avoids the need for specific task preprocessing methods or data expansion techniques. The results of five-fold cross-validation demonstrate the generalization ability of our method, with classification metrics including accuracy, AUC, precision, recall, and F1 score all exceeding 91%. These results underscore the efficacy of our model in aiding radiologists in diagnosing pulmonary tuberculosis images. In our research, we aim to achieve comparable performance using lightweight networks such as MobileNetV2, enabling computation on mobile devices for added convenience in assisting radiologists with diagnosis.

2.1.1 LIMITATIONS

Dependency on Manual Interpretation: Despite the proposed model's advancements in automated tuberculosis detection, it still relies on manual interpretation of chest X-rays by trained radiologists to provide ground truth labels for training data. This dependency may introduce subjective biases and errors into the dataset, affecting the model's performance.

Complexity of Image Interpretation: While the addition of a coordinate attention mechanism enhances the model's ability to focus on relevant features in tuberculosis images, the complexity of image interpretation remains a challenge. The model may struggle with subtle variations and abnormalities in chest X-rays, leading to misclassifications or false positives.

Generalization Across Datasets: While the model demonstrates promising performance on the Shenzhen Third Hospital dataset, its generalization across diverse datasets from different healthcare institutions and geographic regions is uncertain. Variations in image acquisition protocols, patient demographics, and disease manifestations may impact the model's effectiveness in real-world clinical settings.

Need for Further Validation and Clinical Testing: Despite the promising results obtained through cross-validation, further validation and clinical testing are necessary to assess the real-world efficacy. Clinical validation studies involving diverse patient populations and collaboration with healthcare institutions are essential steps towards ensuring the model's clinical utility and safety.

Potential Ethical and Regulatory Considerations: Deployment of automated tuberculosis detection systems in clinical practice raises ethical and regulatory considerations regarding patient privacy, data security, and medical liability. Along with transparent reporting of model limitations and uncertainties, is crucial for responsible implementation and adoption in healthcare settings.

CHAPTER 3

PROPOSED SYSTEM

In this system, detecting the category by which the tuberculosis classification belongs to, the Deep Learning algorithm of Convolutional Neural Networks (CNN) is used. In this proposed system, a pipeline process is proposed for tuberculosis detection which uses deep learning techniques. These systems designed to classify the conditions of tuberculosis and normal from the dataset images with the help of the classification system. To improve detection accuracy, bone images given as input to this project. This system classifies the bone dataset images of the normal and abnormal and also contains two classes conditions. Using the feature extraction method detects the features from the tuberculosis image can be identified and calculate it as feature vectors [4]. In the feature, vectors are used to train and test the convolutional neural network model. Beyond that, tuberculosis classification occurs by using normal and abnormality detection.

Tuberculosis (TB) is one of the most important public health problems worldwide. There are 9 million new TB cases and nearly 2 million TB deaths each year. Case-finding and the management of pulmonary tuberculosis is an essential target of tuberculosis control programs. However, pulmonary tuberculosis (PT) is becoming more and more of a serious problem, particularly in countries affected by epidemics of human immunodeficiency virus (HIV)-TBco-infection. The diagnosis of PT using prompt and accurate methods is a crucial step in the control of the occurrence and prevalence of TB.

Digital image processing uses digital images and computer algorithms to enhance, manipulate or transform images to obtain the necessary information and make decisions accordingly.

Examples of digital image processing include improvements and analysis of the images of the Surveyor missions to the moon , magnetic resonance imaging scans of the brain and electronic face recognition packages. These techniques can be used to assist humans with complex tasks and make them easier. A detailed analysis of an X-ray can help a radiologist. Large number of patients with TB infections needs to be X rayed and screen for active TB to ensure a proper treatment of their infections. Taking Standard Chest X-rays (CXRs) is an inexpensive way to screen for the presence of TB. The purpose of screening system is to identify everything that is or could be related to a patient having TB infections. But mass screening of a large population is a time consuming and tedious work, which require considerable effort when done manually. For this reason, Computer-aided diagnostic systems (CAD) used to detect Tuberculosis infections in chest X rayed. These systems have the potential to lessen the TB detection error risk and also depend on the radiologists. Computer aided detection diagnosis is also known as computer aided diagnosis which is a medical image processing for diagnosingthe aid of radiology images. The goal of CAD is not only to get the better result of the diagnostic accuracy but also the consistent radiology image interpretation by using the computer output. CAD output is able to help the radiologists can diagnosis and provide treatment based on the subjective judgment. Normally, there are two types of approaches are applied in computerized schemes for CAD which are finding the location of lesions and quantifying the features of radiography images whether normal or abnormal patterns are included in lungs. Other medical imaging techniques such as X-ray, ultrasound, etc. produce the information and aid to analyze the detection of tuberculosis in the short time duration.

In this project, we describe how we differentiate between normal and abnormal CXRs with manifestations of TB, using image processing techniques in MATLAB.

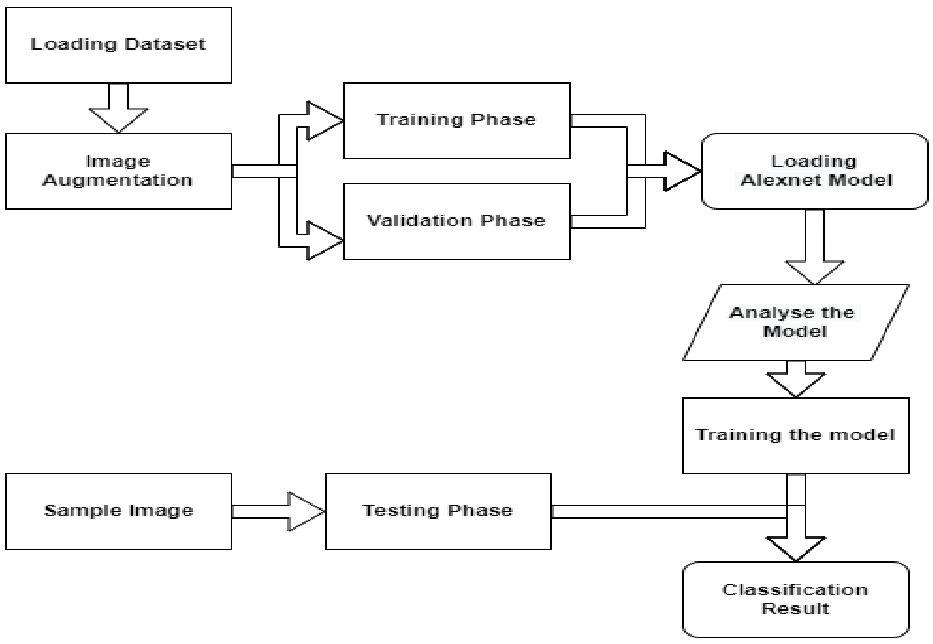


Figure 3.1: Working flow of TB classification

3.1 LOADING DATASET

Data collection, augmentation of data using transformations of the image and finally classification of authentication and un authentication using deep CNN. The experiment has been performed on the fingerprint image data sets, collected from different sources publicly available for research such as the Cancer Imaging Archive. Dataset of Tuberculosis and Normal is shown in figure 3.2.

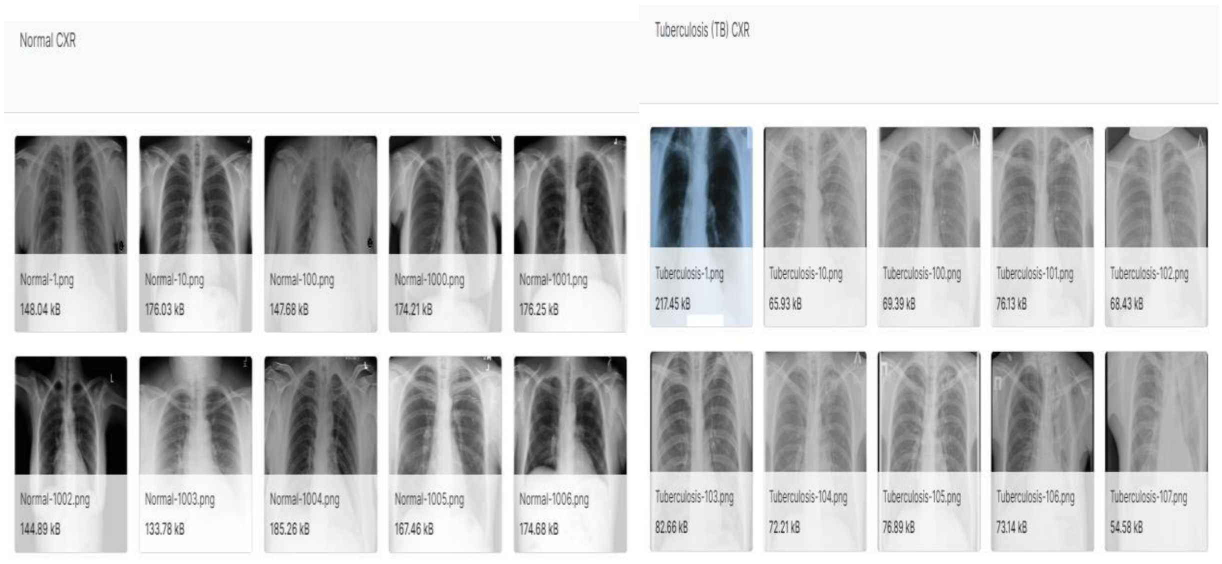


Figure 3.2: Sample image of Tuberculosis and Normal

3.2 IMAGE AUGMENTATION

Image augmentation has one of the main pre-processing is one of the important steps before entering into the training system. It can help in improving the performance of the system. Apart from that, the data augmentation technique also be used. The augmented process, will make the image data-store can be transformed batches of training and test set with optional pre-processing such as resizing, rotation, and reflection. The augmented image data-store is used to resize the training images to 227x227 pixels, which match with the input image of the Google net CNN pre-trained model. The 'R and X Reflection' function is used to flip the train images along the vertical axis randomly, of x and y axis with a pixel range of -20 and 20 is used to translate the training image up to 20 pixels horizontally and vertically, respectively. This data augmentation helps prevent the network from over-fitting and memorizing the exact details of the training images.

3.3 LOADING THE MODEL

In this proposed system, pre-trained CNN is used as a feature extractor and transfer learning by fine-tuning technique is used to transfer the extracted features from the pre-trained CNN model to a new task. In this section, the pre-trained CNN architecture and the fine-tuning pre-trained CNN architecture is described in detail.

3.4 PRE-TRAINED CNN MODEL AS FEATURE EXTRACTOR

Google net pre-trained CNN model is selected as the feature extractor in this proposed system. Google net has been trained on more than a million images from the ImageNet database and it can classify images into 1000 object categories with about 60 million parameters. The architecture of Google net consists of eight learned layers, five convolution layers followed by three fully connected layers. The eight layers Google net architecture is shown in Figure 3.3 below. In MATLAB platform, Google net architecture consists of 25 layers: The first 23 layers are for feature extraction, whereas the last three layers are for classifying these features into 1000 classes. Google net has an image input size of 227x227x3 images with 'zero center' normalization. Then, the first convolution layer filters the 227x227x3 image input image with 96 kernels of size 11×11×3 with a stride of 4 pixels and followed by ReLU non-linear activation, cross channel normalization with five channel per element, and3x3 max pooling with the stride of 2 and zero padding layer. For the second convolution layer, 256 feature kernels with a size of 5x5x48 filters 27x27x96 feature image and carry out further feature extraction. Same as the first convolution layer, second convolution layer also followed by ReLU nonlinearity activation, cross channel normalization with 5 channels per element, and a 3x3 max pooling with the stride of 2 and zero padding and output of 13x13x256 image.

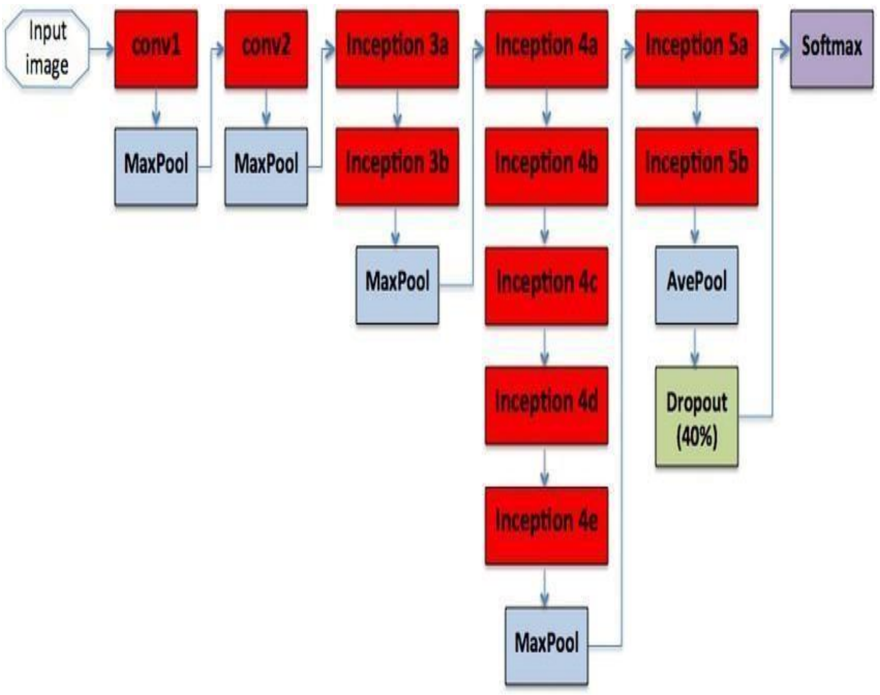
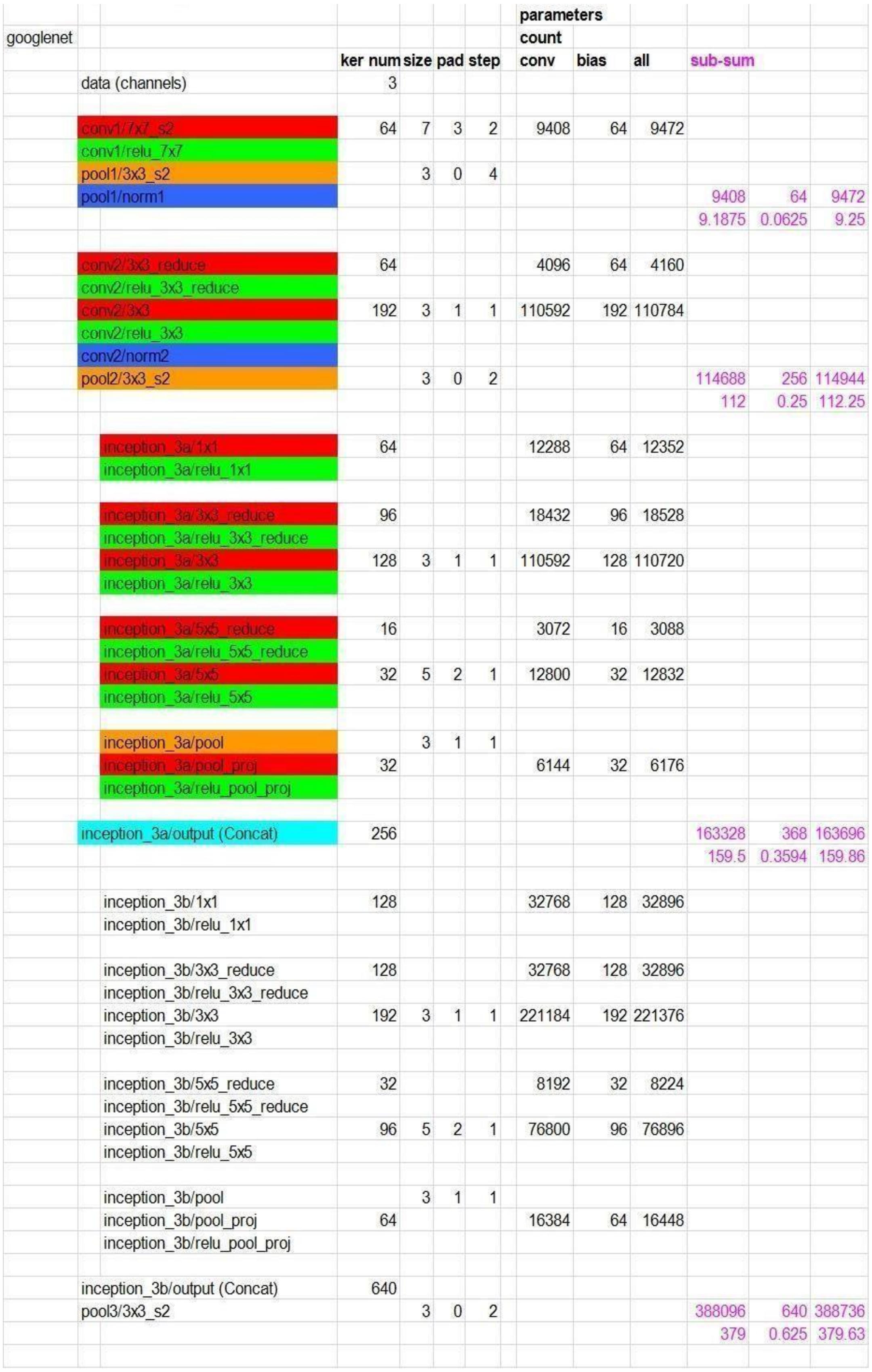


Figure 3.3: Architecture Classification

GoogleNet is a convolutional neural network that is 22 layers deep. You can load a pretrained version of the network trained on either the ImageNet or Places 365 data sets. The network trained on ImageNet classifies images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.



						parameters					
googlenet						count					
		ker num	size	pad	step	conv	bias	all	sub-sum		
	data (channels)	3									
	conv1/7x7_s2	64	7	3	2	9408	64	9472			
	conv1/relu_7x7										
	pool1/3x3_s2		3	0	4						
	pool1/norm1								9408	64	9472
									9.1875	0.0625	9.25
	conv2/3x3_reduce	64				4096	64	4160			
	conv2/relu_3x3_reduce										
	conv2/3x3	192	3	1	1	110592	192	110784			
	conv2/relu_3x3										
	conv2/norm2										
	pool2/3x3_s2		3	0	2				114688	256	114944
									112	0.25	112.25
	inception_3a/1x1	64				12288	64	12352			
	inception_3a/relu_1x1										
	inception_3a/3x3_reduce	96				18432	96	18528			
	inception_3a/relu_3x3_reduce										
	inception_3a/3x3	128	3	1	1	110592	128	110720			
	inception_3a/relu_3x3										
	inception_3a/5x5_reduce	16				3072	16	3088			
	inception_3a/relu_5x5_reduce										
	inception_3a/5x5	32	5	2	1	12800	32	12832			
	inception_3a/relu_5x5										
	inception_3a/pool		3	1	1						
	inception_3a/pool_proj	32				6144	32	6176			
	inception_3a/relu_pool_proj										
	inception_3a/output (Concat)	256							163328	368	163696
									159.5	0.3594	159.86
	inception_3b/1x1	128				32768	128	32896			
	inception_3b/relu_1x1										
	inception_3b/3x3_reduce	128				32768	128	32896			
	inception_3b/relu_3x3_reduce										
	inception_3b/3x3	192	3	1	1	221184	192	221376			
	inception_3b/relu_3x3										
	inception_3b/5x5_reduce	32				8192	32	8224			
	inception_3b/relu_5x5_reduce										
	inception_3b/5x5	96	5	2	1	76800	96	76896			
	inception_3b/relu_5x5										
	inception_3b/pool		3	1	1						
	inception_3b/pool_proj	64				16384	64	16448			
	inception_3b/relu_pool_proj										
	inception_3b/output (Concat)	640									
	pool3/3x3_s2		3	0	2				388096	640	388736
									379	0.625	379.63

Figure 3.4: Details of Googlenet Layers in MATLAB Platform

CHAPTER 4

DESIGN METHODOLOGY

4.1 CONVENTIONAL NEURAL NETWORKS

A CNN contains a sequence of convolutional and max-pooling layers, activation layer and each layer has connected with its previous layer. It is a general, hierarchical feature extractor that will map input image pixel intensities into a feature vector. This was classified by several fully connected layers in the next step [8]. All adjustable parameters are optimized by minimizing the misclassification by reducing the error over the training set. Each convolutional layer performs a 2D convolution with a filter of different size 3 x 3, 5 x 5, 7 x 7. The subsequent activations of the output maps are given by the total of the past convolutional responses which are gone through a nonlinear activation function. Max pooling layer was performing the dimensionality reduction. The output of a thin-layer was given by the most extreme activation over non-covering rectangular areas. Max-pooling makes location invariance and down-samples the image along every direction over a bigger neighborhood. Filter size of convolutional and max-pooling layers are selected in such a way that a fully connected layer can combine the output into a one-dimensional vector. The last layer always be a fully connected layer which contains one output unit for all classes. Here rectification linear unit was used as the activation function. Furthermore, it was deciphered as the likelihood of a specific input image having a place with that class. Adam optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.

4.2 IMAGE CLASSIFICATION OF CNN

Image Classification was the process of finding instances of real-world objects such as faces, buildings, and bicycles in images or videos. Thus, the working process of bone detection and classification based on CNN is shown in Figure 4.1. Image classification algorithms typically use extracted features and learning algorithms to recognize instances of an object category. It was commonly used in applications such as image retrieval, security and advanced driver assistance systems.

Classification of image was an important topic in artificial vision systems and had drawn a significant amount of interest over the last decades. After that, feature detection methods in built on this pre-defined model to training and testing of the our project. By using those features extraction methods some parameters values of features are calculated. However, when a lot of images were given, it was too difficult problem to find features from it. This was one of the reasons that a deep neural network model is used. To extract the features from Alex net that are trained on bone dataset. CNN uses over each image and adjust the kernel as per the propagation in the network.

A kernel was then convolved over the entire image to produces feature maps. As the layers become deeper, the network acquires the knowledge of larger feature extraction. The initial layers take care of the smaller details of the image and the deeper layer can identify the more number details from the image. It was pre-process the images and extract the features by feed-forwarding through the network, and also specify the layer names that can be extracted and save them.

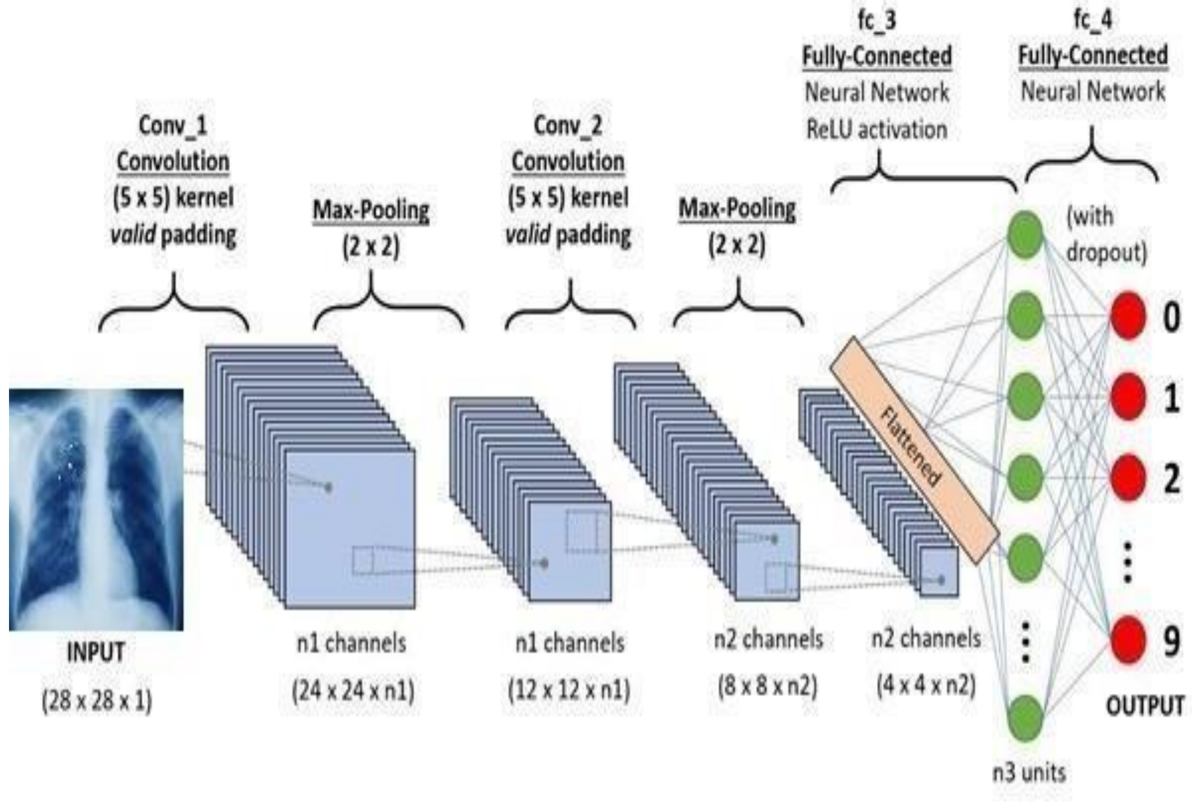


Figure 4.1: CNN Classification

In a Convolutional Neural Network (CNN) for image classification, the input is typically a 28x28 grayscale image represented as a matrix of pixel values. The network starts with Conv_1, a convolutional layer with a (5x5) kernel and valid padding, which produces n1 channels resulting in a (24x24xn1) output. This is followed by a Max-Pooling layer with a (2x2) kernel to down sample the features to (12x12xn1). Subsequently, Conv_2 applies another (5x5) kernel with valid padding, generating n2 channels and outputting features in an (8x8xn2) format. After another Max-Pooling step, the features are further reduced to (4x4x2) with dropout. Finally, the network transitions to fully-connected layers, starting with fc_3 and ReLU activation, leading to fc_4 with n3 units for the final classification output. In the described CNN architecture for image classification, the convolutional layers Conv_1 and Conv_2 extract hierarchical features from the input image, gradually reducing spatial dimensions while increasing the number of channels to capture.

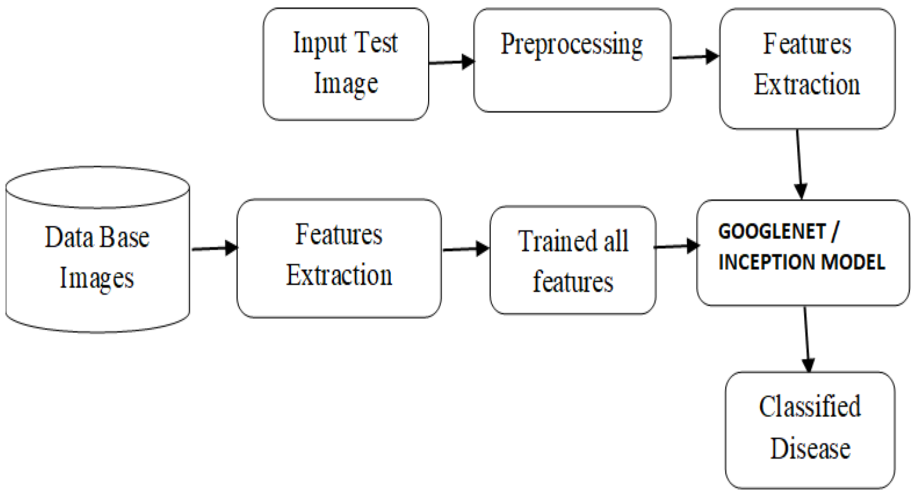


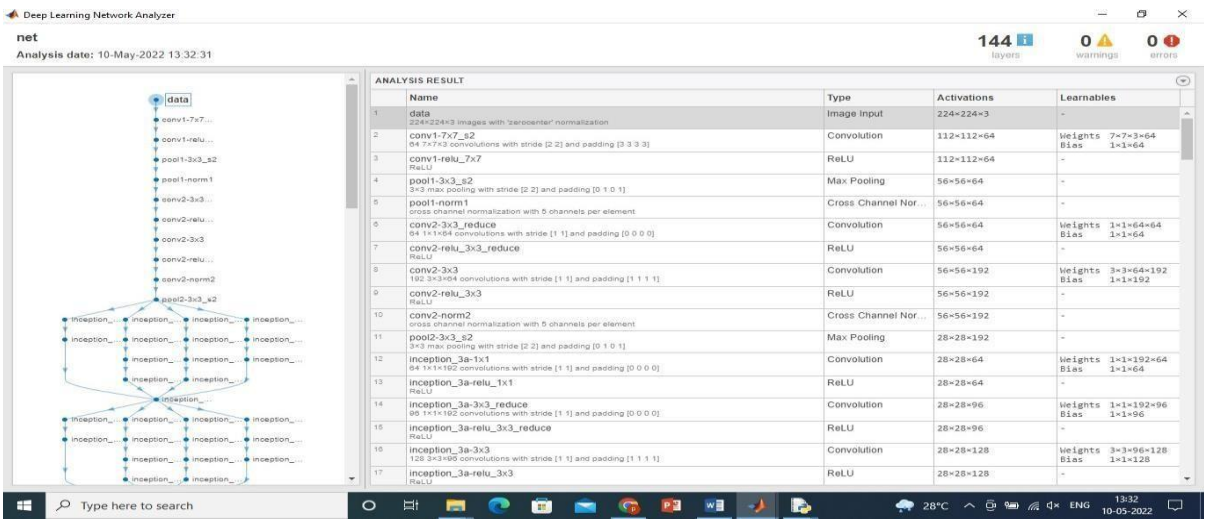
Figure 4.2: Block Diagram

In this above block diagram, before feeding the input image into the CNN, it typically undergoes preprocessing steps. Preprocessing may involve resizing the image, normalizing pixel values, or applying filters to enhance features relevant for classification. This step ensures that the input data is in a suitable format for the CNN model. GoogleNet and Inception are both well-known CNN architectures that have been proven effective for image classification tasks. They consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers. These architectures are capable of capturing complex hierarchical features from images, making them suitable for disease classification tasks. Once the model is trained, it can be used to classify new test images. The trained CNN takes the preprocessed test image as input and outputs the predicted disease class. This classification enables diagnosis or further analysis based on the identified disease or condition.

This could have resulted in the overfitting of the data. The problem of overfitting was solved by modulating the entropic capacity of the network. The amount of information stored in the model. A model that can store a lot of information had that potential to be more accurate but it also ends up storing irrelevant features. In this case, it used a very small CNN with few layers and few filters per layer alongside data augmentation and dropout of 0.5. Dropout also helps reduce overfitting, by preventing a layer from seeing twice the same pattern. Thus, the model which stores less information ends up storing just the most significant features found in the data and was liked to more relevant and generalize better. From the figure 4.4 Network Analysis of Google net Model.

4.2.1 Transfer Learning

Transfer learning is commonly used in deep learning applications. You can take a pre trained network and use it as a starting point to learn a new task. Fine-tuning a network with transfer learning is usually much faster and easier than training a network with randomly initialized weights from scratch. In this project quickly transfer learned features to a new task using a smaller number of training images.


Deep Learning Network Analyzer

net

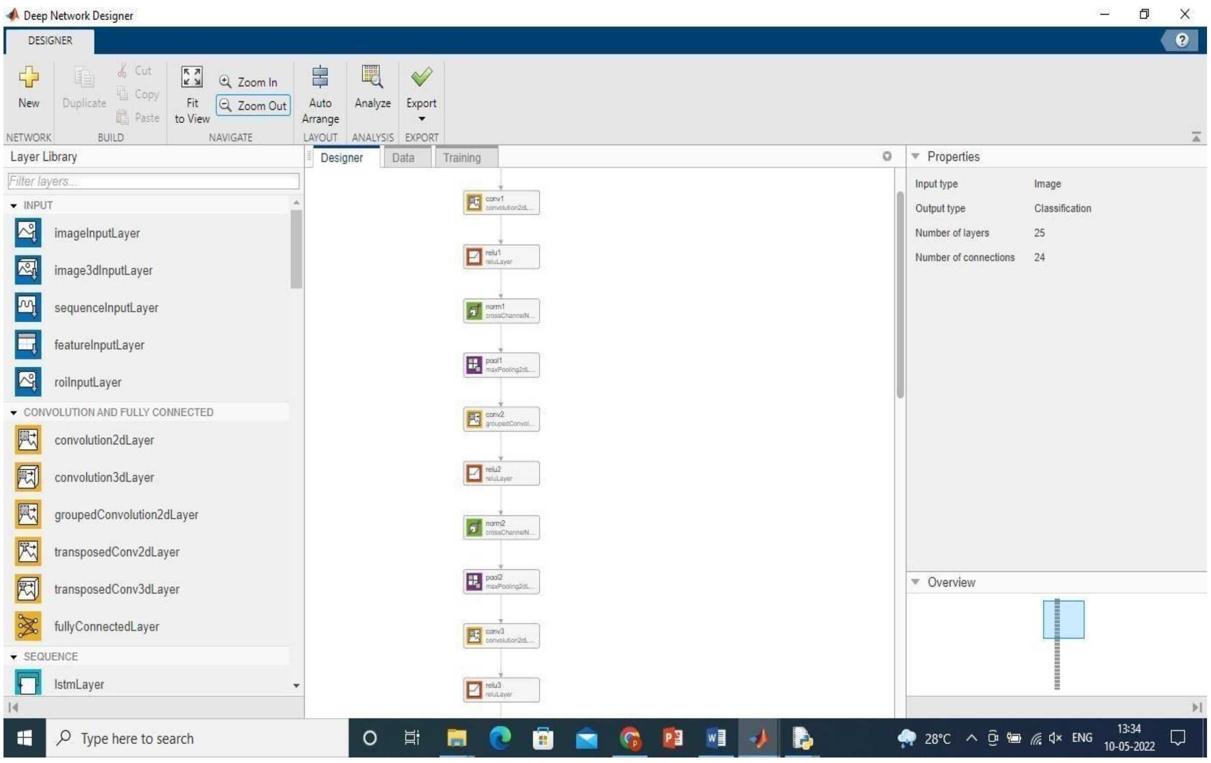
Analysis date: 10-May-2022 13:32:31

144 layers 0 warnings 0 errors

ANALYSIS RESULT

	Name	Type	Activations	Learnables
1	data 224×224×3 images with 'zerocenter' normalization	Image Input	224×224×3	-
2	conv1-7x7_s2 64 7×7×3 convolutions with stride [2 2] and padding [3 3 3 3]	Convolution	112×112×64	Weights 7×7×3×64 Bias 1×1×64
3	conv1-relu_7x7 ReLU	ReLU	112×112×64	-
4	pool1-3x3_s2 3×3 max pooling with stride [2 2] and padding [0 1 0 1]	Max Pooling	56×56×64	-
5	pool1-norm1 cross channel normalization with 5 channels per element	Cross Channel Nor...	56×56×64	-
6	conv2-3x3_reduce 64 1×1×64 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	56×56×64	Weights 1×1×64×64 Bias 1×1×64
7	conv2-relu_3x3_reduce ReLU	ReLU	56×56×64	-
8	conv2-3x3 192 3×3×64 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	56×56×192	Weights 3×3×64×192 Bias 1×1×192
9	conv2-relu_3x3 ReLU	ReLU	56×56×192	-
10	conv2-norm2 cross channel normalization with 5 channels per element	Cross Channel Nor...	56×56×192	-
11	pool2-3x3_s2 3×3 max pooling with stride [2 2] and padding [0 1 0 1]	Max Pooling	28×28×192	-
12	inception_3a-1x1 64 1×1×192 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	28×28×64	Weights 1×1×192×64 Bias 1×1×64
13	inception_3a-relu_1x1 ReLU	ReLU	28×28×64	-
14	inception_3a-3x3_reduce 96 1×1×192 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	28×28×96	Weights 1×1×192×96 Bias 1×1×96
15	inception_3a-relu_3x3_reduce ReLU	ReLU	28×28×96	-
16	inception_3a-3x3 128 3×3×96 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	28×28×128	Weights 3×3×96×128 Bias 1×1×128
17	inception_3a-relu_3x3 ReLU	ReLU	28×28×128	-

Figure 4.3: Loading of Google net architecture



Input type	Image
Output type	Classification
Number of layers	25
Number of connections	24

Figure 4.4: Network Analysis of Googlenet Model

The pre-trained GoogleNet CNN model is utilized as a feature extractor, trained on a large dataset of diverse images. GoogleNet architecture consists of multiple layers, including convolutional layers for feature extraction, followed by fully connected layers for classification. The fine-tuning technique is applied to adapt the pre-trained model to the specific task of TB detection. This involves adjusting the model's parameters through additional training on the TB dataset, allowing it to learn TB-specific features. Through this methodology, the project aims to develop a robust TB detection system capable of accurately classifying CXR images as normal or abnormal, thereby aiding in the early diagnosis and management of TB cases.

CHAPTER 5

REQUIREMENTS

5.1 HARDWARE REQUIREMENT

CPU (Central Processing Unit):

While not as critical as the GPU, a decent CPU is still required for tasks such as data preprocessing, managing training 0pipelines, and handling inference. A multi-core CPU with sufficient RAM is desirable.

Memory (RAM):

Sufficient RAM is necessary, especially when working with large datasets. The size of the dataset, batch size during training, and complexity of the neural network architecture will influence the RAM requirements.

Storage (SSD or HDD):

Fast storage is essential for storing datasets, model checkpoints, and intermediate results during training. Solid-state drives (SSDs) provide faster read/write speeds compared to traditional hard disk drives (HDDs), which can improve overall training performance.

Cooling:

Deep learning tasks can be computationally intensive and can lead to high GPU temperatures. Proper cooling solutions such as fans or liquid cooling might be necessary, especially if you're running models for extended periods.

Power Supply:

Ensure that your system has a stable power supply to support the demanding computational requirements of deep learning tasks, particularly when using GPU accelerators.

5.2 SOFTWARE REQUIREMENT

MATLAB can be a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB was an interactive system whose basic data element is an array that does not require dimensioning. This allows to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar no interactive language such as C or FORTRAN.

5.2.1 MATLAB SYSTEM

The MATLAB system consists of five main parts:

Development Environment: This is the set of tools and facilities that helps use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, and browsers for viewing help, the workspace, files, and the search path.

The MATLAB Mathematical Function Library: This can be a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

The MATLAB Language: This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

Handle Graphics: This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow to fully customize the appearance of graphics as well as to build complete graphical user interfaces on MATLAB applications.

The MATLAB Application Program Interface (API): This is a library that allows to write C and FORTRAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

5.2.2 DEVELOPMENT ENVIRONMENT

MATLAB Desktop

When start MATLAB, the MATLAB desktop appears, containing tools (graphical user interfaces) for managing files, variables, and applications associated with MATLAB. The first time MATLAB starts, the desktop appears as shown in the following illustration, although Launchpad may contain different entries.

Desktop Tools

This section provides an introduction to MATLAB's desktop tools. MATLAB functions to perform most of the features found in the desktop tools. The tools are:

Command Window

Use the Command Window to enter variables and run functions and M-files.

Command History

Lines enter in the Command Window are logged in the Command History window. In the Command History, view previously used functions, and copy and execute selected lines. To save the input and output from a MATLAB session to a file, use the diary function.

Running External Programs

External programs from the MATLAB Command Window. The exclamation point character is a shell escape and indicates that the rest of the input line is a command to the operating system. This is useful for invoking utilities or running other programs without quitting MATLAB. When quit the external program, the operating system returns control to MATLAB.

Launchpad

MATLAB's Launchpad provides easy access to tools, demos, and documentation.

Help Browser

Use the Help browser to search and view documentation for all Math Works products. The Help browser is a Web browser integrated into the MATLAB desktop that displays HTML documents.

Help Navigator

Use to Help Navigator to find information. It includes:

Product filter - Set the filter to show documentation only for the products.

Contents tab - View the titles and tables of contents of documentation for products.

Index tab - Find specific index entries (selected keywords) in the MathWorks documentation for products.

Search tab - Look for a specific phrase in the documentation. To get help for a specific function, set the Search type to Function Name.

Find a term in the page - Type a term in the Find in page field in the toolbar and click Go. Other features available in the display pane are: copying information, evaluating a selection, and viewing Web pages.

Current Directory Browser

MATLAB file operations use the current directory and the search path as reference points. Any file wants to run must either be in the current directory or on the search path.

Search Path

To determine how to execute functions call, MATLAB uses a search path to find M-files and other MATLAB-related files, which are organized in directories on file system. Any file wants to run in MATLAB must reside in the current directory or in a directory that is on the search path. By default, the files supplied with MATLAB and Math Works toolboxes are included in the search path.

Workspace Browser

The MATLAB workspace consists of the set of variables (named arrays) built up during a MATLAB session and stored in memory. Add variables to the workspace by using functions, running M-files, and loading saved workspaces.

To view the workspace and information about each variable, use the Workspace browser, or use the functions. To delete variables from the workspace, select the variable and select Delete from the Edit menu. Alternatively, use the clear function.

Array Editor

Double-click on a variable in the Workspace browser to see it in the Array Editor. Use the Array Editor to view and edit a visual representation of one or two-dimensional numeric arrays, strings, and cell arrays of strings that are in the workspace.

Editor/Debugger

Use the Editor/Debugger to create and debug M-files, which are programs write to run MATLAB functions. The Editor/Debugger provides a graphical user interface for basic text editing, as well as for M-file debugging.

5.2.3 GUI

A graphical user interface (GUI) is a user interface built with graphical objects, such as buttons, text fields, sliders, and menus. In general, these objects already have meanings to most computer users. For example, when move a slider, a value changes; when press an OK button, settings are applied and the dialog box is dismissed. leverage is built-in familiarity must be consistent in how use the various GUI-building components.

The sections that follow describe how to create GUIs with MATLAB. This includes laying out the components, programming them to do specific things in response to user actions, and saving and launching the GUI. In in other words, the mechanics of creating GUIs. This document does not attempt to cover the "art" of good user interface design, which is an entire field unto itself. Topics covered in this section include:

Creating GUIs with GUIDE

MATLAB implements GUIs as figure windows containing various styles of control objects. Program each object to perform the intended action when activated by the user of the GUI. Besides, it must be able to save and launch GUI. All of these tasks are simplified by GUIDE, MATLAB's graphical user interface development environment.

GUI Development Environment

The process of implementing a GUI involves two basic tasks:

- Laying out the GUI components
- Programming the GUI components

GUIDE primarily is a set of layout tools. However, GUIDE also generates an M-file that contains code to handle the initialization and launching of the GUI. This M-file provides a framework for the implementation of the callbacks - the functions that execute when users activate components in the GUI.

Features of the GUIDE-Generated Application M-File

GUIDE simplifies the creation of GUI applications by automatically generating an M-file framework directly from layout. This framework to code application M-file. This approach provides several advantages:

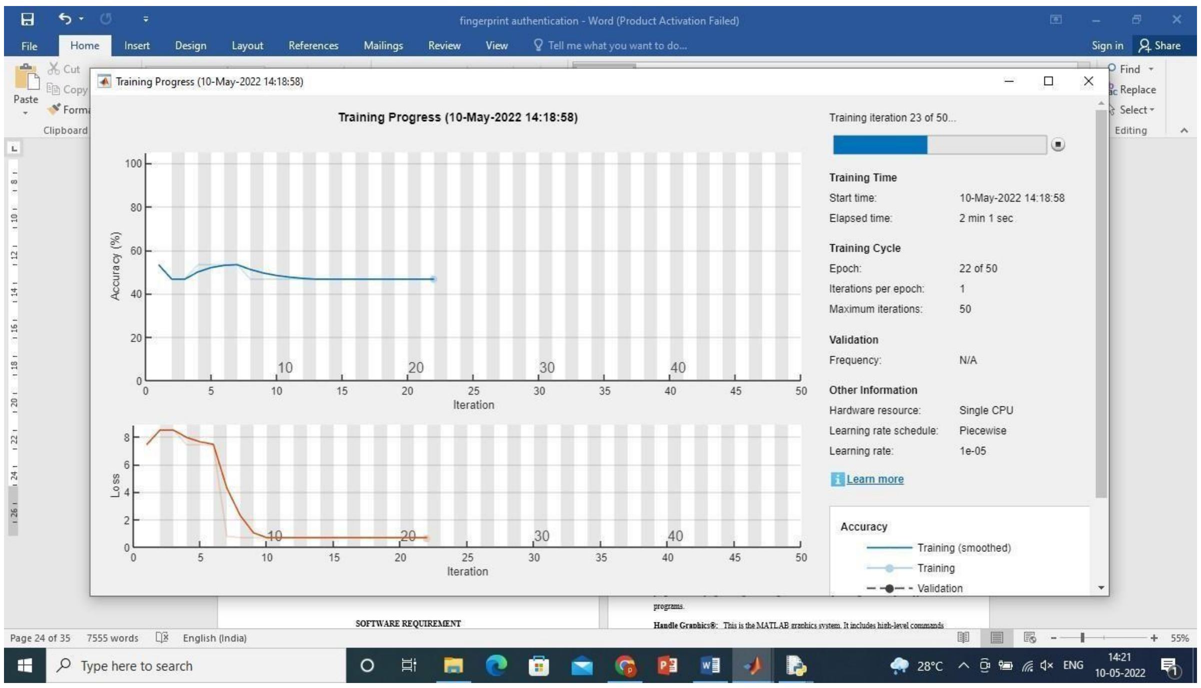
The M-file contains code to implement some useful features (see Configuring Application Options for information on these features). The M-file adopts an effective approach to managing object handles and executing callback routines. The M-files provides a way to manage global data.

CHAPTER 6

TRAINING AND TESTING

6.1 TRAINING NETWORKS

After the network has been structured for classification application with all parameters, then it was ready for training. After each iteration, the network converges by reducing the error rate. The loop was terminating when it reached a minimum error rate. A learning rate was maintained for each network weight (parameter) and separately adapted as learning unfolds. The network weight was adjusted subsequently in each iteration from initial value based on result until it converges to a value. Weight will decide the convergence. The weight value for each image is recorded in a neural network after database loaded. Here learning rate is 0.0001. Those defined weights was further used to classify more number of datasets. Training of authentication and un authentication images on CNN can be done on the given figure 6.1.


fingerprint authentication - Word (Product Activation Failed)
Training Progress (10-May-2022 14:18:58)
Accuracy (%)
Loss
Iteration
Accuracy
Training (smoothed)
Training
Validation

Training iteration 23 of 50...

Training Time
Start time: 10-May-2022 14:18:58
Elapsed time: 2 min 1 sec

Training Cycle
Epoch: 22 of 50
Iterations per epoch: 1
Maximum iterations: 50

Validation
Frequency: N/A

Other Information
Hardware resource: Single CPU
Learning rate schedule: Piecewise
Learning rate: 1e-05

Figure 6.1: Training of TB on CNN

The pre-trained weight which was obtained from the training phase also used in the testing phase. The input image was allowed to pass through all layers of the neural network and parameters were obtained [4]. These values were cross-checked with the pre-trained weight and identify the one which gives maximum matching with the classes presents in the dataset. The system was considering the label to which it is closely matched.

6.2 UML DIAGRAM

The Unified Modelling Language (UML) is used to specify, visualize, modify, construct and document the artefacts of an object-oriented software intensive system under development. Figure 6.2 shows UML offers a standard way to visualize a system's architectural blueprints, including elements such as: Actors, business processes, (logical) components activities programming language statements database schemas, and Reusable software components.

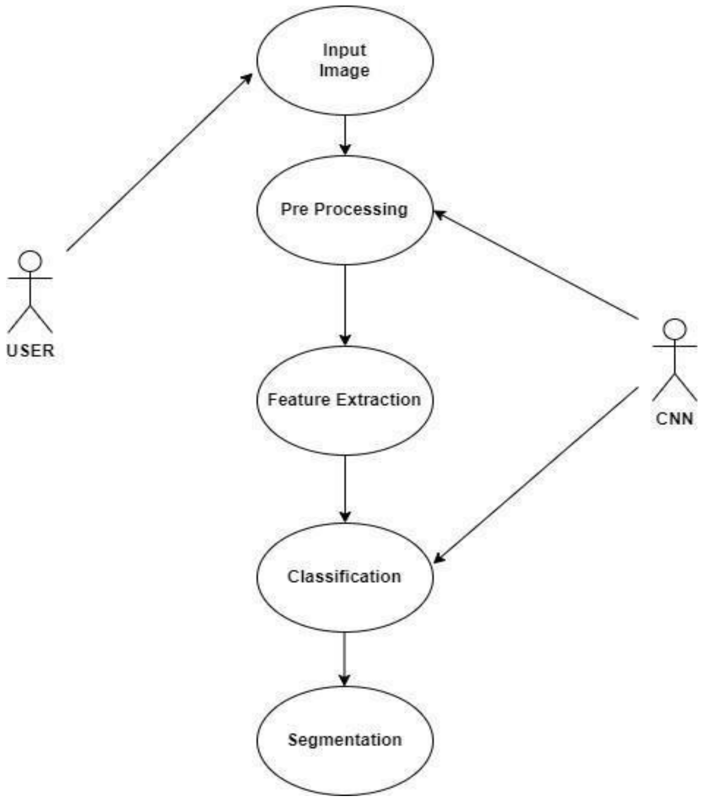


Figure 6.2: UML Diagram for TB Detection

6.3 SEQUENCE DIAGRAM

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communication with its environment. Figure 6.3 shows A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

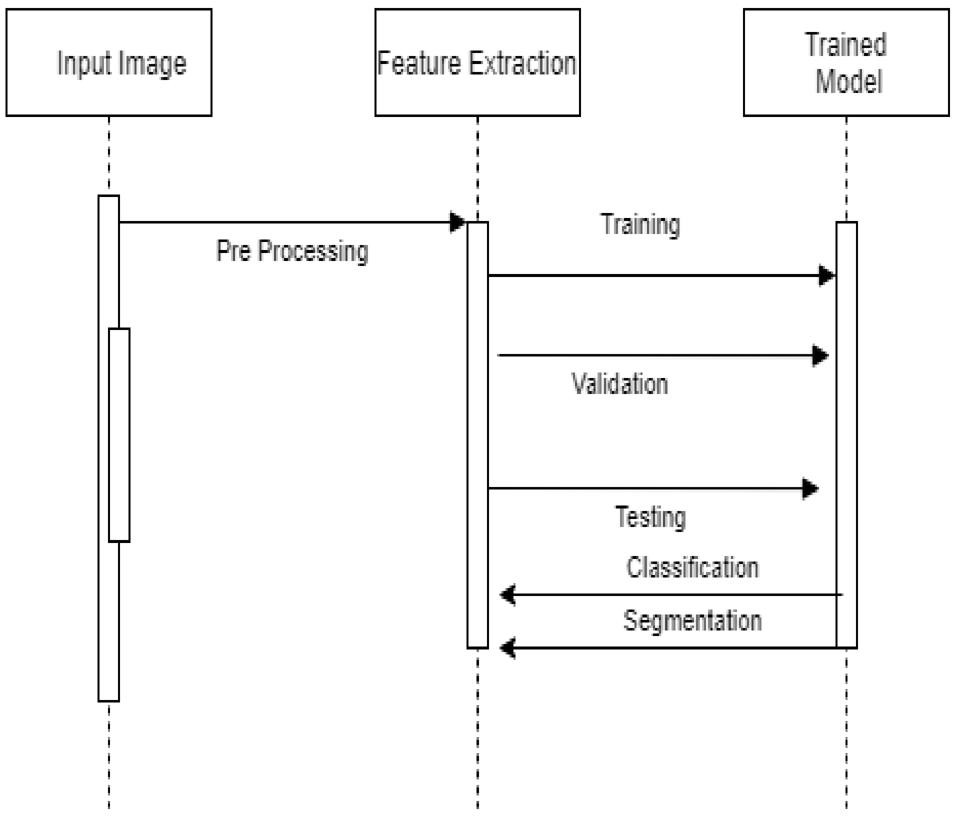
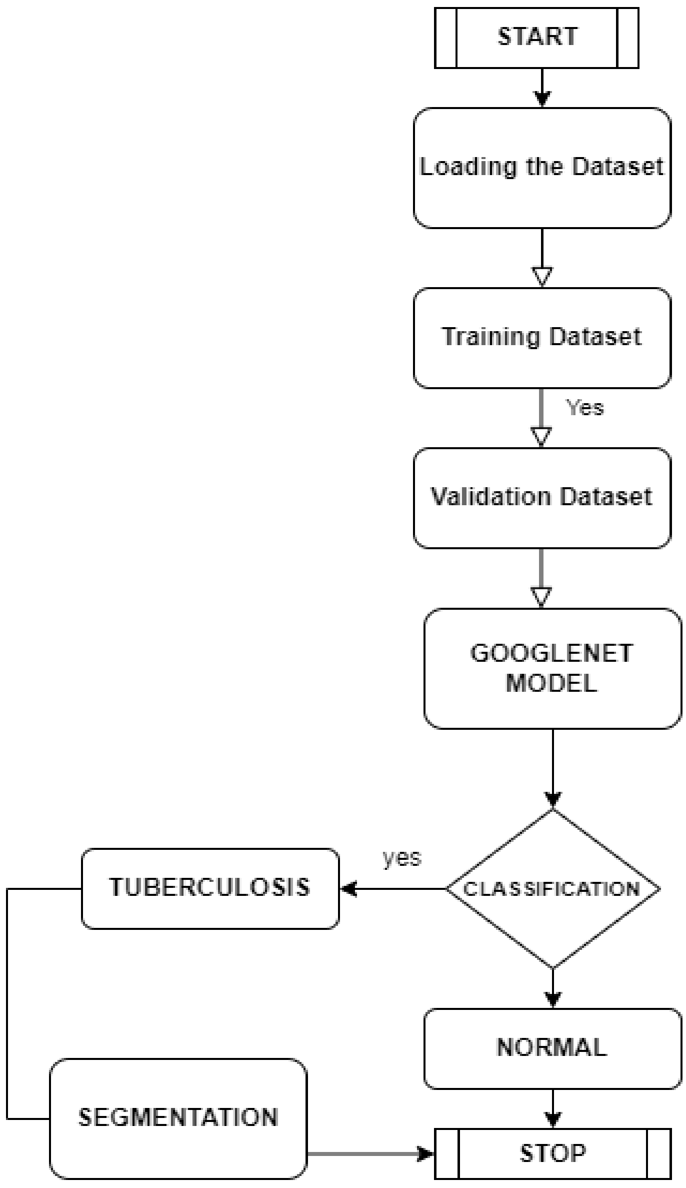


Figure 6.3: Sequence Diagram for TB Detection

6.4 FLOWCHART DIAGRAM

Sequence Flow chart diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. Figure 6.4 shows that the overall flow of control.



-

Figure 6.4: Flowchart for TB Detection

CHAPTER 7

PERFORMANCE MEASURES

Tuberculosis detection refer to the measures used to evaluate the effectiveness and accuracy of algorithms or systems designed to identify Tuberculosis from chest X ray imaging data scans. These metrics are crucial for assessing the reliability and efficiency of the detection process. To evaluate the deep learning models and analyze their performances, some metrics such as the accuracy, recall, precision and F1 score.

Accuracy

Accuracy represents the percentage of correctly identified cases of Tuberculosis among all infants examined, offering a straightforward evaluation of the model's effectiveness. Accuracy measures the number of correct predictions divided by the total number of samples. Applying the equation below, accuracy can be calculated, providing a measure of the model's performance.

Accuracy = ((Number of Correct Predictions)/ (Total Number of Predictions))×100%

To find Number of Correct predictions:

Number of Correct Prediction = Accuracy × Total Number of Predictions

Recall

Recall, specifically in the realm of Tuberculosis detection, refers to the model's capability to correctly identify all actual cases of Tuberculosis among infants from the entire set of brain tumor cases present. It focuses on avoiding missing any positive instances as false negatives can have severe consequences in medical diagnosis, particularly for vulnerable populations like infants.

The formula for recall is below,

$$Recall=TP/(TP+FN) \times 100$$

where:

TP = True positive

TN = True negative

FN = False negative

FP = False positive

Precision

Precision, in the domain of infant Tuberculosis detection, indicates the accuracy of the model in identifying true cases of Tuberculosis among infants from all instances predicted as positive. It focuses on minimizing false positives, ensuring that identified cases are genuinely to prevent unnecessary concern or treatment.

The formula for precision is below,

$$Precision=TP/(TP+FP) \times 100$$

F1 Score

The F1 score is the harmonic mean of precision and recall, and it provides a balance between the two measures. The F1 score is a combined metric that balances both precision and recall, providing a single value to assess the overall performance of a model in infant Tuberculosis detection. It considers both false positives and false negatives and is particularly useful when there's an imbalance between positive and negative cases. The formula for the F1 score as below,

$$F1score=2\times Precision+Recall/Precision\times Recall \times 100$$

It plays a crucial role in evaluating the performance of any proposed solution. In the case of classifying Tuberculosis disease using CNN, the results obtained from the model can be analyzed and compared with the ground truth data to evaluate its performance. The following parameters can be used to evaluate the performance of the model:

Accuracy: It measures the number of correct predictions made by the model out of the total number of samples. The higher the accuracy, the better the model is at classifying the diseases.

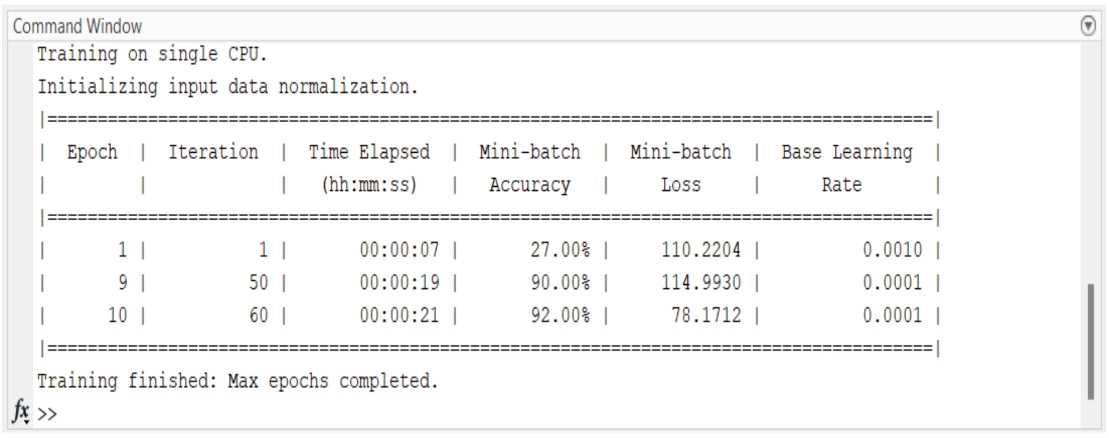
Precision: Precision measures the number of true positive predictions made by the model divided by the sum of true positive and false positive predictions.

Recall: Recall measures the number of true positive predictions made by the model divided by the sum of true positive and false negative predictions.

Table 7.1 Training Configuration Table

Configuration	**Existing Value**	**Proposed Value**
Batch Size	43	20
Learning Rate	0.0001	0.01
Drop Factor	0.6	0.1
Max Epoch	8	10
Step per Epoch	50	100
L2 Regularization	0.00005	0.003

7.1 TRAINING DATA SET

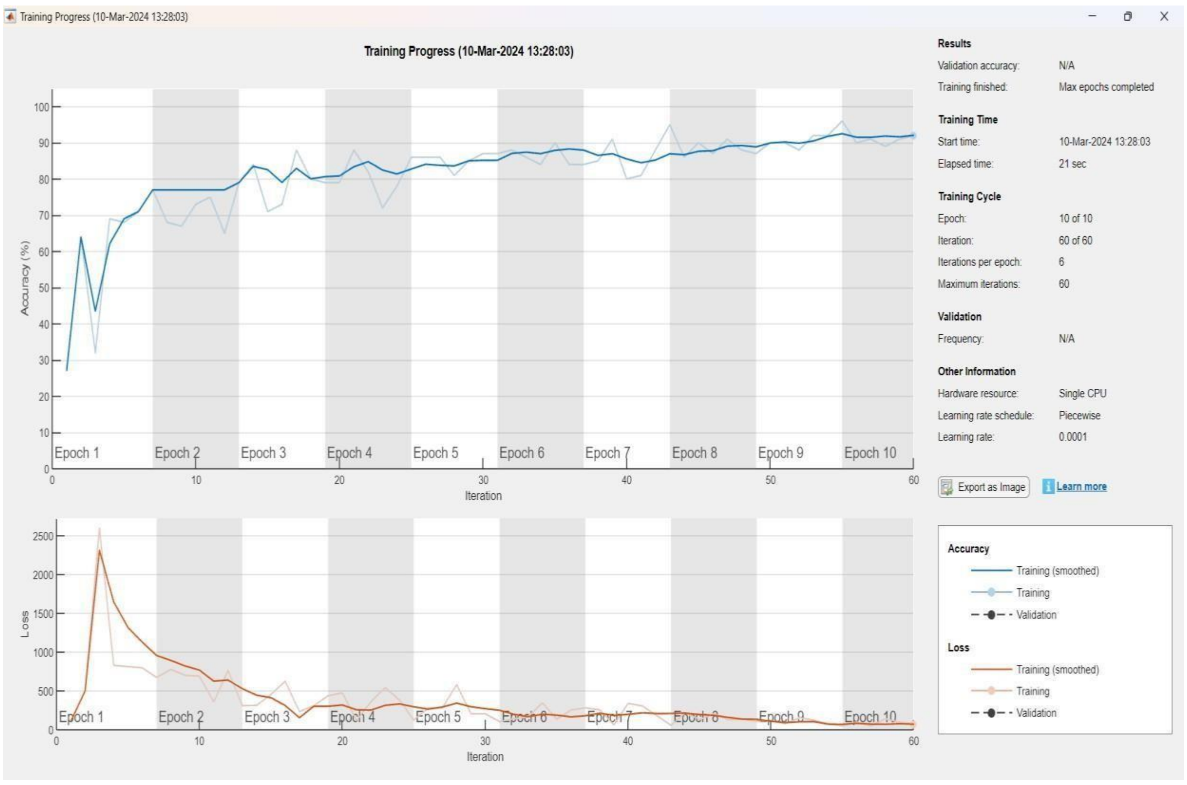


```
Training on single CPU.
Initializing input data normalization.
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|========================================================================================|
|       1 |           1 |       00:00:07 |       27.00% |     110.2204 |          0.0010 |
|       9 |          50 |       00:00:19 |       90.00% |     114.9930 |          0.0001 |
|      10 |          60 |       00:00:21 |       92.00% |      78.1712 |          0.0001 |
|========================================================================================|
Training finished: Max epochs completed.
fx >>
```

Figure 7.1: Training for accuracy & loss

The training dataset is a crucial component in machine learning, including Convolutional Neural Networks (CNNs). It consists of a large set of labeled examples used to train the model. Each example in the training dataset comprises input data (such as images) paired with corresponding target labels (classifications). During the training process, the CNN learns to recognize patterns and features in the input data that correlate with the provided labels. A diverse and representative training dataset is essential for the CNN to generalize well to unseen data and accurately classify new instances. Proper preprocessing, augmentation, and balancing techniques applied to the training dataset can enhance the model's performance and robustness.

7.2 ACCURACY AND LOSS



Results

Validation accuracy:	N/A
Training finished:	Max epochs completed

Training Time

Start time:	10-Mar-2024 13:28:03
Elapsed time:	21 sec

Training Cycle

Epoch:	10 of 10
Iteration:	60 of 60
Iterations per epoch:	6
Maximum iterations:	60

Validation

Frequency:	N/A

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Piecewise
Learning rate:	0.0001

Figure 7.2 Training Progress

Tabel 7.2 Performance on Tuberculosis Detection

REFERENCE	MODEL	ACCURACY (%)	RECALL (%)	PRECISION (%)	F1 SCORE (%)
Proposed	GOOGLE NET	92.00	89.11	89.98	90.00
Tianhao Xu., Zhenming Yuan [9]	VGG16	88.73	85.83	89.73	87.82
Roman Sergeevich Sergeev., Ivan S.Kavaliou.,Uladzislau.[7]	SNP	83.73	88.83	84.73	89.82

The Tuberculosis image dataset is analyzed. In table 7.2, the metrics present the performance of the models with respect to the accuracy, recall,precision and F1 score function results. After evaluating the performance of the Google Net, VGG16, and SNP clustering models, it’s observed that Google Net outperformed the other deep learning models across all metrics, with the highest accuracy, recall, precision, and F1 score.

7.3 NORMAL TB IMAGE

A "normal" output in tuberculosis detection signifies that the chest X-ray analyzed by the system reveals no evidence of active tuberculosis infection or characteristic abnormalities associated with the disease. This classification indicates that the scanned individual's lung structure appears healthy and does not exhibit any typical features indicative of tuberculosis pathology. Such a result holds paramount importance for medical professionals as it confirms the absence of active tuberculosis, guiding treatment decisions, and offering reassurance to patients. It aids in ruling out tuberculosis infection, facilitating timely intervention and appropriate management strategies, ultimately contributing to improved patient outcomes and public health.

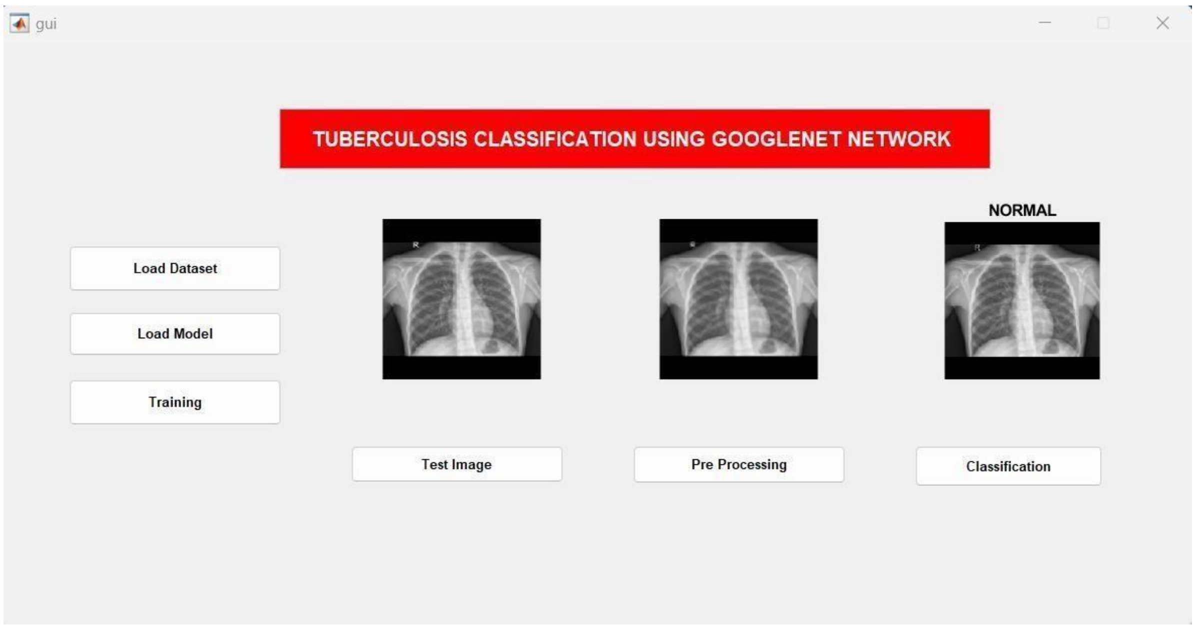


Figure 7.3: Normal Image

Figure 7.3 shows the input image is process into preprocessing and filtering with median filtration which extract the feature under the condition of Googlenet or Inception net and get the classifications as “NORMAL”.

7.4 AFFECTED TB IMAGE

An "abnormal" classification in tuberculosis detection indicates that the chest X-ray being analyzed exhibits abnormalities or irregularities suggestive of active tuberculosis infection or related pulmonary pathology. This outcome alerts medical professionals to the potential presence of tuberculosis-related abnormalities, prompting further diagnostic procedures or investigations to confirm the diagnosis and assess the extent of the infection. The identification of such abnormalities through image analysis is crucial for early detection and intervention, facilitating timely medical management and treatment planning. It enables healthcare providers to initiate appropriate interventions promptly, including the administration of anti-tuberculosis medications and infection control measures, thereby contributing to improved patient outcomes and public health.

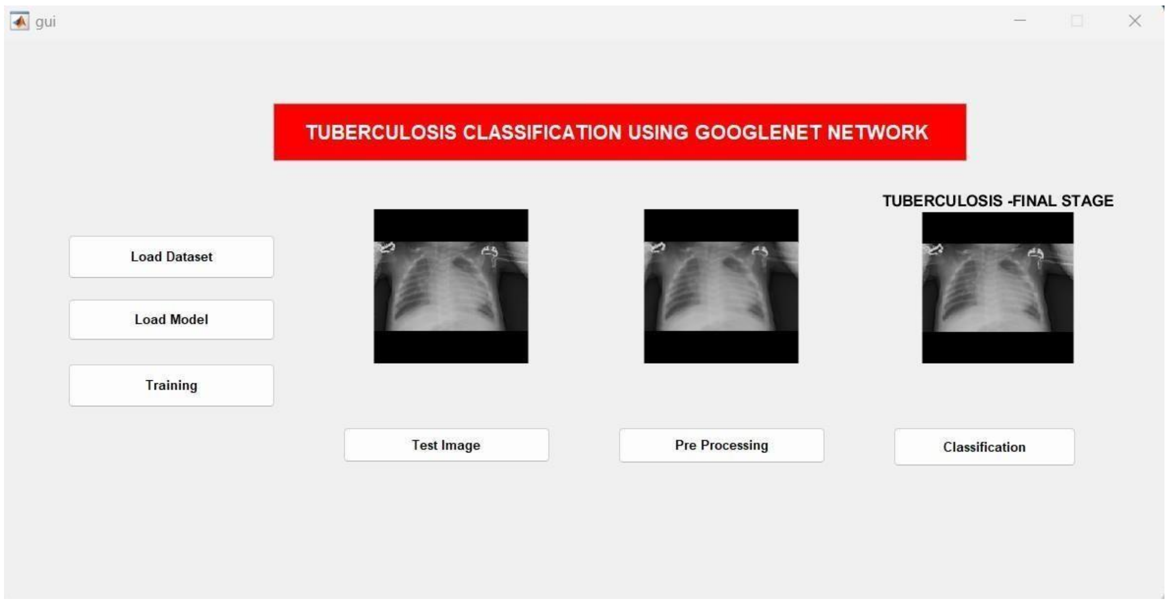

gui
TUBERCULOSIS CLASSIFICATION USING GOOGLENET NETWORK
TUBERCULOSIS -FINAL STAGE
Load Dataset
Load Model
Training
Test Image
Pre Processing
Classification

Figure 7.4: Affected TB Image

Figure 7.4 shows the input image is process into preprocessing and filtering with median filtration which extract the feature under the condition of Google net or Inception net and get the classifications as “ABNORMAL” and also it mentioned the types of stages.

In this testing of normal and tuberculosis sample images for the classification on pre trained model with the prediction accuracy value of normal is 1 with no loss value and prediction accuracy value of normal and tuberculosis is 0.8 with 0.2 was prediction loss.

CHAPTER 8

CONCLUSION & FUTURE ENHANCEMENT

8.1 CONCLUSION

In this project, transfer learning method used for X Ray tuberculosis classification on normal and abnormal detection. X Ray dataset was taken from the clinical diagnostic for normal and abnormal tuberculosis. Different methodologies proposed by various researchers are considered, all of which show that image processing had a major role in chest detection, but no one touches tuberculosis classification. From the performance criteria such as accuracy, loss and this method had been recommended to increase the prognosis. Real-time Application based categorization was one of the main factors in the selection of the technique. Diagnosing tuberculosis abnormalities was a complex and sensitive task to preciseness, reliability. Experiments shows the effectiveness of data augmentation, especially in the case of insufficient training data. In conclusion, this project leveraged transfer learning methodologies for tuberculosis classification in chest X-rays, filling a gap in existing literature that predominantly focuses on image processing for chest abnormalities rather than tuberculosis specifically. Through collaboration with medical experts, the method's clinical relevance was emphasized. Experimentation on diverse datasets validated its robustness across varied populations and settings, while an assessment of computational efficiency highlighted its feasibility for real-time applications. Furthermore, exploration of interpretability techniques provided insights into the model's decision-making process, enhancing trust and transparency. Moving forward, future research directions include investigating novel image features and incorporating multimodal data for improved diagnostic accuracy, ultimately aiming to make a significant clinical impact in tuberculosis diagnosis and management.

8.2 FUTURE ENHANCEMENT

There are opportunities for further improvement for this project from both technical and clinical point of view. For instance, on the technical side, adding segmentation constrain to the method when it goes to abnormal condition. Also extend work for various network model for providing optimum results. Furthermore, training and testing on rigidly aligned images might provide more accurate localization. In clinical application, this proposed method will help the patients can easily understand tuberculosis with module of hardware.

In the pursuit of advancing tuberculosis (TB) detection methods, future work on this project aims to explore various avenues for improvement from both technical and clinical perspectives. This includes integrating multimodal data sources such as clinical history and laboratory results to enhance the robustness of the TB detection system. Additionally, investigating transfer learning techniques and implementing explainable AI methods will contribute to faster convergence, better generalization, and increased interpretability of the model's predictions. Evaluating the system on diverse patient populations and conducting rigorous clinical validation studies will ensure its reliability and effectiveness across different demographic profiles and healthcare settings. Ultimately, these efforts aim to facilitate the seamless integration of the proposed TB detection system into clinical practice, thereby improving patient care and contributing to the fight against tuberculosis.

ANNEXURE

SAMPLE CODE

```
function function varargout = gui(varargin)
% GUI MATLAB code for gui.fig
%      GUI, by itself, creates a new GUI or raises the existing
%      singleton*.
%
%      H = GUI returns the handle to a new GUI or the handle to
%      the existing singleton*.
%
%      GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI.M with the given input arguments.
%
%      GUI('Property','Value',...) creates a new GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before gui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to gui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui

% Last Modified by GUIDE v2.5 03-Jul-2020 14:56:25
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui_OpeningFcn, ...
                   'gui_OutputFcn',  @gui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before gui is made visible.
function gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui (see VARARGIN)

% Choose default command line output for gui
```

```
handles.output = hObject;

a=ones([256 256]);
axes(handles.axes1);
imshow(a);

axes(handles.axes2);
imshow(a);

axes(handles.axes3);
imshow(a);

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global train
matlabpath
='C:\Users\vigne\OneDrive\Desktop\tubercolsis\Tubercolsis_Network_Model_Go
oglenet'
data = fullfile(matlabpath,'dataset')


train = imageDatastore(data, 'IncludeSubfolders',true,'LabelSource','foldernames');
count = train.countEachLabel;


msgbox('Dataset Loaded Successfully')


% Update handles structure
guidata(hObject, handles);


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global layers
disp('Pre-Trained Model Loaded...')
net = googlenet;
analyzeNetwork(net);
layers = [ imageInputLayer([200 200 3])


net(2:end-3)
fullyConnectedLayer(3)

softmaxLayer

classificationLayer()

]
msgbox('Pre-Trained Model Loaded Successfully')

% Update handles structure
guidata(hObject, handles);

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%training
global opt training train layers
opt = trainingOptions('adam', ...
    'InitialLearnRate', 0.001, ...
    'LearnRateSchedule', 'piecewise', ...
    'LearnRateDropFactor', 0.1, ...
    'LearnRateDropPeriod', 8, ...
    'L2Regularization', 0.004, ...
    'MaxEpochs', 10, ...
    'MiniBatchSize', 100, ...
    'Verbose', true, 'Plots','training-progress');
```

```
training = trainNetwork(train,layers,opt);

msgbox('Trained Completed')

% Update handles structure
guidata(hObject, handles);

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double



% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global inp
cd input
[file path] = uigetfile('*.bmp;*.png;*.jpg','Pick an Image File');


if isequal(file,0)
       warndlg('File not selected');
else

inp = imread(file);
cd ..
axes(handles.axes1);
imshow(inp);
img=inp;
if size(inp,3)>1
    Freg = rgb2gray(inp);
end
handles.img=img;
end
% Update handles structure
guidata(hObject, handles);
```

```
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global inp J
img = rgb2gray(inp);
J = medfilt2(img);
axes(handles.axes2);
title('Filtered Image');
imshow(J);
% Update handles structure
guidata(hObject, handles);
% --- Executes on button press in pushbutton4.

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global training inp out
out = classify(training,inp);
axes(handles.axes3);
imshow(inp);
title(string(out));
% Update handles structure
guidata(hObject, handles);
```

REFERENCE

[1] Ayaz. M, Shaukat. F, Raja. G. “Ensemble learning based automatic detection of tuberculosis in chest X-ray images using hybrid feature descriptors”, Phys Eng Sci Med. (2021).

[2] Bakyarani. E; Srimathi. H; Arul Leena Rose. PJ. "A Comparative Study On Performance Of Pre-Trained Convolutional Neural Networks In Tuberculosis Detection". European Journal of Molecular & Clinical Medicine, 7, 3, 2020, 4852-4858).

[3] Becker. A, “Detection of tuberculosis patterns in digital photographs of chest X-ray images using deep learning: Feasibility study”. Int. J. Tuberc. Lung Dis. 22, 328–335 (2018).

[4] Christodoulidis. S, Anthimopoulos. M, Ebner. L, Christe. A, and Mougiakakou. S, ‘‘Multisource transfer learning with convolutional neural networks for lung pattern analysis,’’ IEEE J. Biomed. Health Inform., vol. 21, no. 1, pp. 76–84, Jan. (2017).

[5] Chowdhury. M. H, Shuzan. M. N. I, Chowdhury. M. E. H, Mahbub. Z. B,Uddin. M. M, Khandakar. A, and Reaz. M. B. I, ‘‘Estimating blood pressure from the photoplethysmogram signal and demographic features using machine learning techniques”, Sensors, vol. 20, no. 11, p. 3127, Jun. (2020).

[6] Chhikara. P, Singh. P, Gupta. P, and Bhatia. T, ‘‘Deep convolutional neural network with transfer learning for detecting pneumonia on chest X-rays” (2020).

[7] Degnan. A. J, Ghobadi. E. H, Hardy. P, Krupinski. E, Scali. E. P, Stratchko. L, Ulano. A, Walker. E, Wasnik. A. P, and Auffermann. W. F, ‘‘Perceptual and interpretive error in diagnostic radiology—Causes and potential solutions”, (2019).

[8] Devasia. J, Goswami. H, Lakshminarayanan. S, Rajaram. M and Adithan.S, "Deep learning classification of active tuberculosis lung zones wise manifestations using chest X-rays: A multi label approach", vol. 13, (2023).

[9] Guo. R, Passi. K & Jain. C. K. “Tuberculosis diagnostics and localization in chest X-rays via deep learning models”, Front. Artif. Intell. 3, 583427 (2020).

[10] Hwang. EJ, Park. S, Jin. KN, Kim. JI, Choi. SY, Lee. JH, Goo. JM, Aum J, “Development and Validation of a Deep Learning-based Automatic Detection Algorithm for Active Pulmonary Tuberculosis on Chest Radiographs”. Clin Infect Dis (2019).

[11] Hernández. A, Panizo. A, and Camacho. D, ‘‘An ensemble algorithm based on deep learning for tuberculosis classification,’’ in Proc. Int. Conf. Intell. Data Eng. Automated Learn., Manchester, U.K., (2019).

[12] Iqbal. A; Usman. M; Ahmed. Z “Tuberculosis chest X-ray detection using CNN-based hybrid segmentation and classification approach”. Biomed. Signal Process. (2023).

[13] Jiang. H.; Diao. Z.; Shi. T.; Zhou. Y.; Wang. F.; Hu. W.; Zhu. X.; Yao. Y.D, “A review of deep learning-based multiple-lesion recognition from medical images: Classification, detection and segmentation”, (2023).

[14] Kandel. I, Castelli. M, “How Deeply to Fine-Tune a Convolutional Neural Network: A Case Study Using a Histopathology Dataset”, Applied Sciences (2020).

[15] Kaul. V, Enslin. S, Gross. S. A, Gozes.O and Greenspan. H, "Deep Feature Learning from a Hospital-Scale Chest X-ray Dataset with Application to TB Detection on a Small-Scale Dataset", (2019).

[16] Kim, S. et al. “Deep learning in multi-class lung diseases’ classification on chest X-ray images Diagnostics”,12, 915 (2022).

[17] Kieu. STH, Bade. A, Hijazi. MHA, Koliv. H, “A Survey of Deep Learning for Lung Disease Detection on Medical Images: State-of-the-Art, Taxonomy, Issues and Future Directions”, (2020).

[18] Mahmood, Ammar & Giraldo, Ana & Bennamoun, Mohammed & An,

Senjian & Sohel, Ferdous & Boussaid, Farid & Hovey, Renae & Fisher, Robert & Kendrick, Gary, “Automatic Hierarchical Classification of Kelps Using Deep Residual Features”, (2020).

[19] Mizan. M. B, Hasan. M. A. M and Hassan. S. R, "A Comparative Study of Tuberculosis Detection Using Deep Convolutional Neural Network," 2020 2ndInternational Conference on Advanced Information and Communication Technology (ICAICT), (2020), pp. 157-161.

[20] Mogaveera. R, Maur. R, Qureshi. Z and Mane. Y, "Multi class chest X-ray classification of pneumonia tuberculosis and normal X-ray images using convnets", Proc. ITM Web Conf., vol. 44, (2022).

[21] Nguyen. Q. H, Nguyen. B. P, Dao. S. D, Unnikrishnan. B, Dhingra. R, Ravichandran. S. R, Satpathy. S, Raja. P. N, and Chua. M. C. H, ‘‘Deep learning models for tuberculosis detection from chest X-ray images”, Apr. (2019).

[22] Pasa. F, Golkov. V, Pfeiffer. F, Cremers. D and Pfeiffer. D, "Efficient deep network architectures for fast chest X-ray tuberculosis screening and visualization", Sci. Rep., vol. 9, no. 1, (2019).

[23] Puttaguntal. M. K and Ravi. S, “Detection of TB based on Deep Learning based methods”, (2021).

[24] Ravi. V; Acharya. V; Alazab. M, “A multichannel Efficient Net deep learning-based stacking ensemble approach for lung disease detection using chest X-ray images”, (2023).

[25] Rustum. R, Babu. G. C, Praveen. P, Lakshmi. V, Indupalli. T and Sowjanya. A, "Detection of Tuberculosis using Machine Learning Techniques and Image Preprocessing", 2022 International Conference on Inventive ComputationTechnologies (ICICT) Nepal, (2022).

[26] Santosh. K, Allu. S, Rajaraman. S and Antani. S, "Advances in deep learning for tuberculosis screening using chest X-rays: The last 5 years review", J. Med. Syst., vol. 46, no. 11, (2022).

[27] Shreyas Rai; Shivangi Singh; Shivansh Nautiyal; Anupkumar

Bongale;Prachi Kadam, “A Comparative Study for Chest X-rays Indicating Tuberculosis”, (2023).

[28] Singh. M, Pujar. G. V and Kumar. S. A, “Evolution of Machine Learning in TB Diagnosis: A Review of Deep Learning-Based Medical Applications”, (2022).

[29] Tawsifur Rahman. T, Amith Khandakar, Muhammad A. Kadir, Khandaker R. Islam, Khandaker F. Islam, Zaid B. Mahbub, Mohamed Arselene Ayari, Muhammad E. H. Chowdhury. "Reliable Tuberculosis Detection using Chest X-ray with Deep Learning, Segmentation and Visualization". IEEE Access, Vol. 8, pp 191586 - 191601. (2020).

[30] Wang. Q, Wu. B, Zhu. P, Zuo. W, and Hu. Q, “Efficient channel attention for deep convolutional neural networks’’,in Proc. IEEE/CVFConf. Comput. Vis. Pattern Recognit. (CVPR), Jun. (2020).

[31] X. et al. “Automated abnormality classification of chest radiographs using deep convolutional neural networks” Npj Digit. Med. 3, 70(2020).

[32] Yadav. O, Passi. K, and Jain. C. K, ‘‘Using deep learning to classify X-ray images of potential tuberculosis patients,’’ in Proc. IEEE Int. Conf. Bio inf. Biomed. (BIBM), Dec. (2018).