

# TUBERCULOSIS DETECTION IN CHEST RADIOGRAPHY

## MINI PROJECT REPORT

***Submitted by***

MOHANASUNDAR S [20AD033]

TAMILARASU A [20AD051]

AYYAPPAN M [20AD007]

***In partial fulfillment for the award of the degree of***

## BACHELOR OF TECHNOLOGY

**IN**

## ARTIFICAL INTELLIGENCE AND DATA SCIENCE

**MUTHAYAMMAL ENGINEERING COLLEGE (AUTONOMOUS)**

## RASIPURAM

## DEC 2022

**MUTHAYAMMAL ENGINEERING COLLEGE (AUTONOMOUS)**

## RASIPURAM

## BONAFIDE CERTIFICATE

Certified that this Report “**TUBERCULOSIS DETECTION IN CHEST RADIOGRAPY”** is the confide work of “**MOHANASUNDAR S [20AD033] TAMILARASU A [20AD051] AYYAPPAN M [20AD007]**” who carried out the work under my supervision.

|  |  |
| --- | --- |
| Signature of the Supervisor  **Mrs .M.S.SOUNDARYA, M.E,**  **ASSISTANT PROFESSOR**  **SUPERVISOR**  Department of Artificial intelligence and data science,  Muthayammal Engineering College(Autonomous),  Rasipuram-637408. | Signature of the HOD  **DR.P.SRINIVASAN M.E.PH.D.,**  **PROFESSOR**  **HEAD OF DEPARTMENT**  Department of Artificial intelligence and data science,  Muthayammal Engineering College(Autonomous),  Rasipuram-637408. |

## Submitted for the Design Project Viva-Voce examination held On \_\_\_\_\_\_

**INTERNAL EXAMINER EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

We heartly thank our Chairman **Shri. R.KANDASAMY,** Muthayammal Educational Trust & Research Foundation.

We express our heartful thanks to **Dr. K.GUNASEKARAN, M.E., Ph.D. ,F.I.E..,** Secretary and Managing trustee for providing necessary facilities.

We thank our beloved Principal **Dr. M.MADHESWARAN, M.E., Ph.D., M.B.A.,** profusely for all the motivation and moral support he had extended to us to complete this project successfully.

We extend our sincere thanks to **Dr.P.SRINIVASAN, M.E., Ph.D.,** Head of the Department of Artificial intelligence and data science for his guidance and kind co- operation.

We are greatly thankful to our Project Coordinator **Mrs.M.S SOUNDARYA., M.E.,** Assistant Professor, Department of Artificial intelligence and data science for her valuable guidance and motivation, which helped us to complete this project on time.

We express our sincere thanks to **Mrs.M.S.SOUNDARYA, M.E.,** Assistant Professor, Department of Artificial intelligence and data science, Project Guide for her valuable guidance in our project and making it success.

We whole heartedly thank all the philanthropic teaching and non-teaching faculty members of our department without whom we would not be able to make this project a reality.

|  |  |  |
| --- | --- | --- |
| **CHAPTER No.** | **TITLE** | **PAGE No.** |
|  | **ABSTRACT** | vi |
|  | **LIST OF ABBREVATIONS** | vii |
|  | **LIST OF FIGURE** | viii |
|  | **LIST OF TABLE** | ix |
| **1** | **INTRODUCTION** | 1 |
| **2** | **LITERATURE REVIEW** | 5 |
| **3** | **FEASIBILITY STUDY** | 8 |
| **4** | **SYSTEM ANALYSIS** | 11 |
|  | 4.1. Existing System | 11 |
|  | 4.2. Proposed System | 11 |
|  | 4.2.1. Expected Merits | 13 |
| **5** | **SYSTEM REQUIREMENTS** | 14 |
|  | 5.1. Hardware Requirements | 14 |
|  | 5.2. Software Requirements | 14 |
|  | 5.3. Software Description | 15 |
|  | 5.3.1. Python | 15 |
|  | 5.3.2 Complex Software  Development | 16 |
|  | 5.3.3 Python Features | 19 |
| **6** | **PROJECT DESCRIPTION** | 25 |
|  | 6.1. System Implementation | 25 |
|  | 6.1.1 Architectural Design | 25 |
|  | 6.2 Image Acquisition | 26 |
|  | 6.2.1 Image pre-processing | 27 |

|  |  |  |
| --- | --- | --- |
|  | 6.3.1 Database Augmentation | 29 |
|  | 6.4 CNN Classification | 32 |
|  | 6.5 Dataset and Experimental Settings | 34 |
|  | SETTINGS | 36 |
| **7** | **DATASET DESCRIPTION** | 38 |
|  | 7.1Dataset U-NET | 38 |
|  | TB Classification | 41 |
|  | SUPPLEMENTARY MATERIALS | 45 |
| **8** | **METHODOLAGY** | 47 |
| **APPENDIX A** | **SOURCE CODE** | 50 |
| **APPENDIX B** | **SCORCE CODE** | 62 |
| **9** | **CONCLUSION & FUTURE**  **ENHANCEMENTS** | 66 |
| **10** | **REFERENCE** | 69 |

# ABSTRACT

Tuberculosis is a major health treat is many region of world . Tuberculosis is a chronic lung disease that occurs due to bacterial infection and one of top 10 leading causes of death. Several public database were used to create a database 700 TB inflected and 1000 normal chest x-ray image for this study. Convolutional Neural Network(CNN) various image preprocessing methods are tested find combination yield highest accuracy. Nine different deep CNNs (ResNet18, ResNet50, ResNet101, ChexNet, InceptionV3, Vgg19, DenseNet201, Squeeze Net, and MobileNet) were used for transfer learning from their pre-trained initial weights and were trained, validated and tested for classifying TB and non-TB normal cases . The proposed method with state-of-the-art performance can be useful in the computer-aided faster diagnosis of tuberculosis. Thus, in the study, we propose an automatic TB detection system using advanced deep learning (DL) models. A significant portion of a CXR image is dark, providing no information for diagnosis and potentially confusing DL models. Therefore, in the proposed system, we use sophisticated segmentation networks to extract the region of interest from multimedia CXRs

**LIST OF ABBREVATIONS**

|  |  |
| --- | --- |
| TB | TUBERCULOSIS |
| CXR | CHEST RADIO GRAPHY |
| CNN | CONVOLUTIONAL NEURAL NETWORK |
| WHO | WORLD HEALTH ORGANIZATION |
| PA | POSTEROANTERIOR |
| NN | NEURAL NETWORK |
| DL | DEEP LEARNING |
| AI | ARTIFICIAL INTELLIGENCE |
| CAD | COMPUTER AIDED DESIGN |
| CAM | COMPUTER AIDED MANIFICATION |
| DBMS | DATABASE MANAGEMENT SYSTEM |
| TDD | TEST DRIVEN DEVELOPMENT |

# 

# LIST OF FIGURE

# 

|  |  |  |
| --- | --- | --- |
| **FIG NO** | TITLE | PAGE NO |
| 1.1 | X-Ray Image Machine | 2 |
| 1.2 | Multi Layer Neural Network | 3 |
| 2.3 | Illustrated Comparison Of Original X-Ray And Photography | 9 |
| 6.1 | Proposed Image Processing Pipeline | 25 |
| 6.1.1 | Score Cam Heat Map On A Chest X-Ray Image | 26 |
| 6.2.1 | Example Of X-Ray Image | 28 |
| 7.1 | Architecture Of U-Net | 39 |
| 7.2.1 | Comparison Of The Roc Curves For Normal And Tb | 44 |
| 7.2.2 | Confusion Matrix For Normal And Tb | 45 |
| 8 | Overview Of The Complete System | 49 |
| 9 | U-Net Output Result For Cxr Image | 67 |

# LIST OF TABLE

|  |  |  |
| --- | --- | --- |
| TABLE NO | TITLE | PAGE NO |
| 6.5.1 | Gather The Normal Data In Dataset | 35 |
| 6.5.2 | Gather The Tuberculosis Data In Dataset | 36 |
| 7.2.1 | Comparison Performance Of Different Models For Tb And Classification Using With And With Out Segmented Data | 43 |
| 9 | Comparison With Other Recent Similar Work | 68 |

## CHAPTER 1

## INTRODUCTION

Tuberculosis (TB) is a potentially serious infectious disease that mainly affects the lungs. Most of the symptoms can be detected by reviewing the Chest X-ray images of the potential patients. Tuberculosis (TB) is a communicable disease caused by a bacterium called Mycobacterium tuberculosis. The digital medical industry has evolved significantly in the past decade. Most institutions are steering away from the analog way of operations and moving to digital. Tuberculosis has visual symptoms like Consolidation, Fibrosis, Infiltration, Mass, Nodule, Pleural Thickening, Pneumonia etc., most of which can be recognized in a Chest Xray image. In clinical practice, chest radiographs are examined by experienced physicians for the detection of TB. However, this is time consuming and a subjective process. Subjective inconsistencies in disease diagnosis from radiograph is inevitable. Importantly,[1] CXR images of tuberculosis are often misclassified to other diseases of similar radiologic patterns. Thus, it is of great importance to detect these symptoms when trying to assess whether the patient is infected with Tuberculosis or not. These symptoms however, usually cover an extremely small region of the entire image. The specific theme investigated in this research paper is to use neural networks to diagnose such images for the existence of tuberculosis bacteria presence in digital x-ray data. With patient data being digital this would allow and open the door to computerized diagnostics and artificial intelligence detection mechanisms. In this perspective, computer aided diagnosis (CAD) systems can play important role in the mass screening of pulmonary TB by analyzing the chest X-ray images. Residual networks a variety of the CNNs were used to overcome the vanishing gradient problem in cases where the region of interest is minute. The dataset used in was organized by segmenting the data into two categories; ‘normal’ and ‘abnormal’. The dataset employed for training the model was asymmetric, i.e.4248 images in the ‘abnormal’ category and 453 images in the ‘normal’ category.



Fig 1.1. X-ray image machine

Several research groups used classical machine learning techniques for classifying TB and non-TB cases from CXR images. CNNs have been used in several recent studies for the detection of lungs diseases including pneumonia and tuberculosis by analysing chest X-ray images. Typically, whole X-ray images were used for the detection of lung disorders using CNN. However, the X-ray images contains lungs as well as other regions of the thorax although the disease like TB is manifested in the lung region only. Nine popular pre-trained deep learning CNNs such as ResNet18, ResNet50, ResNet101 [49], DenseNet201 [50], ChexNet [51], Squeeze Net [52], InceptionV3 [53], VGG19 [54] and MobileNetV2 [55] . If Neural Networks are used to detect the disease with a Radiologist confirming the result, the whole process would be much quicker and efficient. The first being various preprocessing methods and the second the hybrid method. The hybrid method is a combination between Computer Aided Detection (CAD) and DCNN. Both methods show excellent results with the hybrid method showing the highest accuracy. This CNN architecture was shown to have improved performance in detecting normal and abnormal x-rays with an accuracy of 99.8 %However, the X-ray images contains lungs as well as other regions of the thorax although the disease like TB is manifested in the lung region only. Thus, focusing on the lung region of the X-ray images during training and classification may significantly improve the performance of TB detection. To the best of our knowledge, no such work regarding the use of deep learning networks on segmented lungs for TB detection is reported. This paper focuses on the detection of TB using transfer learning based technique of CNNs on the original and segmented lungs in X-ray images.

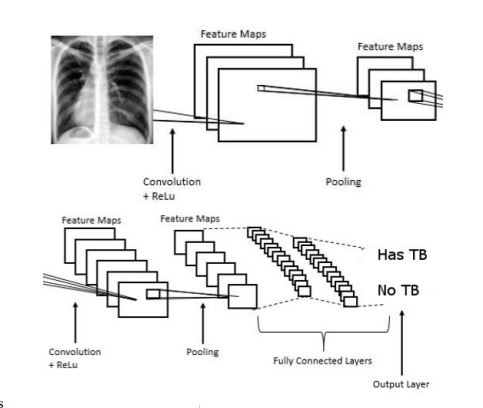


Fig 1.2. Multi Layer Neural Network

## Thus, in the study, we propose an automatic TB detection system using advanced deep learning (DL) models. A significant portion of a CXR image is dark, providing no information for diagnosis and potentially confusing DL models. Therefore, in the proposed system, we use sophisticated segmentation networks to extract the region of interest from multimedia CXRs.

## Several important contributions were reported in this study. The purpose of the proposed work is to develop a novel automatic TB detection system from CXR images using segmentation and DL models to improve the performance across different types of objective and subjective metrics to demonstrate TB detection outcomes. The proposed system will automatically predict the outcome of TB as either positive or negative based on a patient’s CXR. The performance of the system is evaluated on multiple public datasets with both positive and negative TB cases. We also used augmentation rotation in nine different angles in both normal and segmented CXR images. In addition, the proposed system incorporates XAI to visualize prediction outputs to aid medical personnel in TB diagnosis.

## CHAPTER 2

## LITERATURE REVIEW

The datasets used in our study are "China-Shenzhen set Chest X-ray Database" [3,9,13], "Montgomery County Chest X-ray Database" [3,9,13], "NIH Chest X-ray Dataset of 14 Common Thorax Diseases" [15]. The NIH dataset is a large dataset of more than 100,000 medium resolution images. The China and Montgomery dataset combined total to just 800 images. However, It contains very high-resolution pictures of tuberculosis cases and could be accessed from various sources. The China dataset as well as the Montgomery dataset are the databases for Tuberculosis created by National Library of Medicine, Maryland, USA. The NIH dataset is extracted from the clinical PACS database at National Institutes of Health Clinical Center and consists of 112,120 frontal view X-ray images of 30,805 unique patients with the text-mined fourteen disease image labels.

Our second CXR set, the Shenzhen set, is from Shenzhen No.3 Hospital in Shenzhen, Guangdong providence, China. Shenzhen Hospital is one of the largest hospitals in China for infectious diseases, with a focus both on their prevention and treatment. The CXRs we received from Shenzhen Hospital are from outpatient clinics. They were captured within a one month period, mostly in September 2012, as part of the daily routine at Shenzhen Hospital, using a Philips DR Digital Diagnose system. The set contains 340 normal CXRs and 275 abnormal CXRs with TB. For the Shenzhen set, we have the radiologist readings, which we consider as ground-truth.

Diagnostic Standards and Classification of Tuberculosis in Adults and Children. This official statement of the American Thoracic Society and the Centers for Disease Control and Prevention was adopted by the ATS Board of Directors, July 1999. This statement was endorsed by the Council of the Infectious Disease Society of America, September 1999.

Another research found on image processing is found in [9] which like [12–15] helps the early diagnosis of breast cancer using image processing; within this research, we can find that it uses the technique of segmentation by texture; the images used as evidence are from a database, which has images of cancerous masses and microcalcifications manually labelled by experts. In [16–18], they carry out the identification of breast cancer using thermal images, perform a digital processing of the images, using a texture analysis of the images to identify, and extract all the regions of interest.

Image processing is widely used, for example, in [6]; image processing is used to extract regions of interest with properties that can potentially be related to the medical diagnosis of Parkinson’s; it uses computer-aided diagnosis technology to process the images, extract the textures, make a segmentation of the image, and find the area of interest. Within [7], we found the use of image processing, pattern recognition, and artificial intelligence to help detect clusters of microcalcifications in digitized mammography images. Few articles [8, 9] used free databases, making it difficult to compare new techniques and even replicate results. +e first results of their use were presented to the literature after the consolidation of two open and free databases of radiographic images [9] but considering the use of computer vision techniques for lung region segmentation [9]. Although the latter takes a different approach than the one that will be discussed in this paper, it encourages the use of these freely available databases for model training and testing. Most of the studies that looked into the use of multilayer perceptron neural networks for TB detection did not consider the use of medical images to feed such ANNs. On the contrary, they thought that laboratory parameters (cholesterol, creative, blood pressure, amylase level, and so on) and data from office exams (body temperature, cough, and difficulty breathing) could be used to provide experience with ANNs. Although these studies demonstrate the feasibility of using ANNs to detect tuberculosis from real-world data, they require medical examinations and trained professionals to provide input parameters to neural networks, which may not be available or feasible in certain circumstances, particularly given the majority profile of TB patients. To address these limitations, the approach proposed in this paper only uses radiographic images of the lungs, a low-cost and widely available exam that is thus more appropriate for realistic scenarios. Some works of interest where a compendium of techniques to improve medical images can be found are [8–10], and to eliminate noise from the image using techniques that range from erosion, extraction, and others commonly used in the state of the art, they can be referred in [11]

# CHAPTER 3

# FEASIBILITY STUDY

## Patient population

This was a prospective observational study of patients enrolled at the integrated TB-HIV (human immunodeficiency virus) out-patient clinic of the Infectious Diseases Institute in Kampala, Uganda. CXRs were taken in the Radiology Department of Mulago.

National Referral Hospital, which is the main national and teaching hospital providing specialized care to the whole country. The study received ethics approval from the Joint Clinical and Research Centre Ethics Committee, Kampala, and the Uganda National Council for Science and Technology, Kampala (HS 1303).

## Image analysis

# For image analysis, industrial-grade Deep Learning image analysis software (Suite v2.0; ViDi Systems, Villaz-Saint-Pierre, Switzerland) was used. In Deep Learning, artificial neural networks are arranged in multiple layers, which imitates the brain, their natural counterpart. The neurons of the mammalian neocortex are organised in multiple layers, which is particularly apparent in the human visual cortex. This means that any given input percept will be deconstructed and represented at multiple levels of abstraction, each corresponding to a different layer or cortical area .Although currently not approved for routine clinical use, ViDi has recently shown human-like performance in the detection of breast cancer.

# 

# Fig no: 3 Illustrated comparison of original X-ray and photograph. Examples of an original digital chest X-ray image (DCXr, top left) and a digital photograph of a film radiograph (photo, top right).

# Statistical analysis

# Due to the preliminary nature of the present study, the statistical analysis is mainly descriptive. Continuous variables are expressed as mean and standard deviation; categorical variables as counts and percentages. Detection performance was assessed using receiver operating characteristics (ROC) analysis with the area under the curve (AUC). Pathology classification was assessed with a confusion matrix generated using Python 3.5.2, seaborn 0.7.1, numpy 1.12.0 and pandas 0.19.2 packages (Python Software Foundation, Beaverton, OR, USA). The specificity and positive predictive value (PPV) were calculated for each feature in the whole data set as well as for the validation data only

# CHAPTER 4

# SYSTEM ANALYSIS

## Existing System

Tuberculosis (TB) is classified as one of the top ten reasons for death from an infectious agent. Tuberculosis disease infected humans usually mucus on chest and cough and weight loss. Tuberculosis based on the patient chest X-ray images using on Convolutional Neural Networks (CNN). Various image preprocessing methods are tested to find the combination that yields the highest accuracy. Moreover, a hybrid approach using the original statistical computer-aided detection method combined with Neural Networks was also investigated. Simulations have been carried out based on 1000 normal images & 700 abnormal images. The simulations show that a cropped region of interest coupled with contrast enhancement yields excellent results. When further enhancing the images with the hybrid method even better results are achieved.

## Proposed System

Unlike the other existing tuberculosis disease detection methods, we've got developed a model which runs in an exceedingly very less time and also gives more accuracy together with the remedies for the disease detected within the chest x-ray. Convolutional neural networks (CNNs) are multi-layered networks whose architecture determines the performance of the network. It consists of three parts namely, convolution layer, pooling layer and fully connected layer. the primary two together forms the feature extractor and also the third layer acts as a classifier.

The pooling layer reduces the dimensionality of the features extracted by the convolutional layer. The fully connected layer followed by softmax uses the feature extracted to classify the photographs. The convolution layer takes input image and extracts the features employing a set of learnable filters. The real of every filter with the raw image pixel in window manner provides the 2-D feature map. The rectified long measure (ReLU) is one in all the foremost popular activation function. The max pooling layer could be a sub-sampling layer that reduces the dimensions of feature map. Then the fully connected layer provides a full connection to every of the generated feature map. Softmax assigns decimal probabilities to every class in an exceedingly multiclass problem to classify the pictures.

**In the present work, three classification methods were used**:

* The first method used based on support vector machines (SVMs) is a supervised learning model with associated algorithms that analyse the data and recognize patterns. the data points closest to the hyperplane, or the elements of a data set that, if deleted, would change the location of the dividing hyperplane, are called support vectors. As a result, they might be regarded essential components of data collection. SVM stands for support vector machine and is a supervised machine learning technique that may be used for classification and regression. SVMs are more typically utilized in classification issues, thus that is where we will concentrate our efforts in this paper.
* The second method used is based on logistic regression (LR), which is a classification machine learning algorithm that is used to predict the probability of a categorical-dependent variable that is dichotomous; that is, it contains data that can be classified in one of two possible categories (dead or alive, sick or healthy, yes or no, and so on).
* The third method used is based on the nearest neighbors (KNN, K-Neighbors Classifier), which is an algorithm based on supervised type instances of machine learning

## 4.2.1 Expected Merits

1. Collecting the database and segregating it per the features of the human chest x-ray.
2. Taking a picture because the test image/ input image.
3. Training the pictures from datasets for the CNN classification.
4. Comparison of those both test and train images.
5. After comparison, all the features are extracted.
6. Diseases within the plant are found.

**ADVANTAGES:**

* Accuracy level in prediction is improved.
* Large number images are collected and processed to extend accuracy level.

## CHAPTER 5

## SYSTEM REQUIREMENTS

**5.1 HARDWARE REQUIREMENTS:**

The section of hardware configuration is a very important task associated with the software development insufficient random access memory may affect adversely on the speed and efficiency of the complete system. the method should be powerful to handle the complete operations. The magnetic disc should have sufficient capacity to store the file and application.

* + - System : RYZEN 5.
    - Hard Disk : 120 GB.
    - Ram : 1GB.

## 5.2 SOFTWARE REQUIREMENTS:

A serious element in building a system is that the section of compatible software since the software within the market is experiencing in progression. Selected software should be acceptable by the firm and one user additionally because it should be feasible for the system. This document gives an in depth description of the software requirement specification. The study of requirement specification is concentrated specially on the functioning of the system. It allow the developer or analyst to know the system, function to be allotted the performance level to be obtained and corresponding interfaces to be established.

* + - Operating system : Windows 11.
* Coding Language : Python
* Platform: Google Colab
* Database : MYSQL.

## 5.3 SOFTWARE DESCRIPTION:

**5.3.1 Python**

In technical terms, Python is an object-oriented, high-level programing language with integrated dynamic semantics primarily for web and app development. it's extremely attractive within the field of Rapid Application Development because it offers dynamic typing and dynamic binding options. Python is comparatively simple, so it is easy to be told since it requires a singular syntax that focuses on readability. Developers can read and translate Python code much easier than other languages. In turn, this reduces the value of program maintenance and development because it allows teams to figure collaboratively without significant language and knowledge barriers.

Additionally, Python supports the utilization of modules and packages, which implies that programs are often designed in an exceedingly modular style and code are often reused across a spread of projects. Once you've developed a module or package you wish, it will be scaled to be used in other projects, and it is easy to import or export these modules.

One in every of the foremost promising benefits of Python is that both the quality library and therefore the interpreter are available freed from charge, in both binary and source form. there's no exclusivity either, as Python and every one the mandatory tools are available on all major platforms. Therefore, it's a tasty option for developers who don't need to fret about paying high development costs.

If this description of Python over your head, don't be concerned. You'll comprehend it in time. What you would like to require aloof from this section is that Python may be a artificial language wont to develop software on the online and in app form, including mobile. It's relatively easy to be told, and also the necessary tools are available to all or any freed from charge. that produces Python accessible to almost anyone.

**Multiple Programming Paradigms**

Like other modern programming languages, Python also supports several programming paradigm. It supports object oriented and structured programming fully. Also, its language features support various concepts in functional and aspect- oriented programming. At the identical time, Python also features a dynamic type system and automatic memory management. The programming paradigms and language features facilitate your to use Python for developing large and complex software applications.

**Compatible with Major Platforms and Systems**

At present, Python is supports many operating systems. you'll be able to even use Python interpreters to run the code on specific platforms and tools. Also, Python is an interpreted programing language. It allows you to you to run the identical code on multiple platforms without recompilation. Hence, you're not required to recompile the code after making any alteration. The feature makes it easier make changes to the code without increasing development time.

**5.3.2 Complex Software Development**

Python could be a general purpose programing language. Hence, you'll be able to use the artificial language for developing both desktop and web applications. Also, you'll use Python for developing complex scientific and numeric applications. Python is meant with features to facilitate data analysis and visualization. you'll be able to profit of the info analysis features of Python to form custom big data solutions without putting time beyond regulation and energy. At the identical time, the info visualization libraries and APIs provided by Python facilitate your to visualise and present data during a more appealing and effective way. Many Python developers even use Python to accomplish AI (AI) and tongue processing tasks.

**Adopt Test Driven Development**

Use Python to make prototype of the software application rapidly. Also, you'll be able to build the software application directly from the prototype just by refactoring the Python code. Python even makes it easier for you to perform coding and testing simultaneously by adopting test driven development (TDD) approach. you'll be able to easily write the desired tests before writing code and use the tests to assess the appliance code continuously. The tests may also be used for checking if the appliance meets predefined requirements supported its ASCII text file.

However, Python, like other programming languages, has its own shortcomings. It lacks a number of the built-in features provided by other modern artificial language. Hence, you have got to use Python libraries, modules, and frameworks to accelerate custom software development.

Also, several studies have shown that Python is slower than several widely used programming languages including Java and C++. you've got to hurry up the Python application by making changes to the applying code or using custom runtime. But we are able to always use Python to hurry up software development and simplify software maintenance.

**Benefits of Learning Python**

There are many benefits of learning Python, especially as your mother tongue, which we'll discuss. it's a language that's remarkably easy to be told, and it will be used as a stepping stone into other programming languages and frameworks. If you're an absolute beginner and this can be your first time working with any style of coding language, that's something you want.

Python is widely used, including by variety of huge companies like Google, Pinterest, Instagram, Disney, Yahoo!, Nokia, IBM, and plenty of others. The Raspberry Pi - which may be a mini computer and DIY lover's dream - relies on Python as it's main artificial language too. You're probably wondering why either of those things matter, and that is because once you learn Python, never have a shortage of how to utilize the skill. to not mention, since lots of massive companies depend on the language, you'll be able to observe money as a Python developer.

**Other benefits include:**

* Python are often wont to develop prototypes, and quickly because it's very easy to figure with and browse.
* Most automation, data processing, and massive data platforms depend upon Python. this is often because it's the best language to figure with for general purpose tasks.
* Python allows for a more productive coding environment than massive languages like C# and Java. Experienced coders tend to remain more organized and productive when working with Python, as well.
* Python is simple to read, whether or not you are not a talented programmer. Anyone can begin working with the language, all it takes may be a little bit of patience and lots of practice. Plus, this makes it a perfect candidate to be used among multi-programmer and enormous development teams.
* Python powers Django, an entire and open source web application framework. Frameworks - like Ruby on Rails - is accustomed simplify the event process.
* It an enormous support base due to the very fact that it's open source and community developed. many like-minded developers work with the language on a daily and still improve core functionality. the most recent version of Python continues to receive enhancements and updates as time progresses. this can be a good thanks to network with other developers.

**Python Environment Setup**

One in every of the foremost important things you'll do when working with any artificial language is setup a development environment which allows you to execute the code you write. Without this, you may never be able to check your work and see if your website or application is freed from syntax errors.

With Python, you furthermore might need something called an interpreter that converts your code - which makes up the whole lot of your application - to something the pc can read and execute. Without this interpreter, you will have no thanks to run your code. To convert your code, you want to first use a Python shell, which calls upon the interpreter through something called a "bang" line.

**Python Shell versus Text File**

A shell may be a program or tool which will be wont to interact with a system. as an example, the Windows package shell may be tapped into by employing a "terminal" or instruction to submit commands and arguments. With Python, things work a touch differently than an OS shell. The Python shell is employed to interact with an interpreter, which feeds code to a computer in a very form that it can understand.

After you execute a Python program that you have written, the interpreter reads the code and converts it into usable commands. The important thing to notice is that each one of this is often done after the program has been executed. With a shell, the interpreting - or conversion - happens in real-time as you type the code into the pc or system. this implies that the particular program is executing as you type. After you write code in an exceedingly computer file, none of that happens until you feed the document into an interpreter. If you have got Python installed on your computer you'll be able to call upon the interpreter employing a instruction, but this step is completed after you've already written the code.

This makes it harder to identify errors in your code, and it can even be frustrating if the interpreter runs into issues, because they'll not be as apparent as they'd if you had used a shell. Still, lots of developers like better to use a text editing tool because it's simple and straightforward to try to.

**5.3.3 Python Features:**

Python is usually such as Perl, Ruby, PHP, Scheme, and Java. this is often because it's an incredibly powerful object-oriented language. Python also has several notable features which make it a tasty language to figure with for developers.

Python makes use of a sublime syntax, meaning the programs you write are much easier to read. this can be because they're closer to the human language, or how we write our words, rather than a language that computers use to read and interpret code. for instance, the "print" command will display anything proceeding it - and in quotes - at runtime

* + - * Python is easy and easy-to-use, which implies that it's much easier to induce your programs up and running. that's why Python is taken into account ideal for prototype development and similar ad-hoc programming tasks. It doesn't compromise maintainability either.
      * It comes with the quality Python Library, offering integrated support for a spread of common programming tasks like syncing with web servers, ransacking through text, and modifying files. For a majority of other languages, you've got to make this content from scratch.
      * It includes an interactive mode that simplifies testing for brief snippets of code. There's even a development environment bundled with it called IDLE. The dev environment makes setup most easier and faster.
      * The language may be extended by adding new modules, whether or not they have been compiled in C or C++. Even better, the modules are often used as shortcuts in future projects once they have been created.
      * Python may be embedded into an application, which can provide a programmable interface for users of that app. this is often an excellent feature if you're building an app that may teach coding, or requires working with Python during a terminal.
      * It is compatible with a protracted list of computers and operating systems like Windows, Linux, MacOS, many brands of Unix, OS/2, and more. Furthermore, it uses an identical interface on each of these platforms, which suggests you'll be able to jump between them easily if necessary.
      * It is truly free because it doesn't cost anything to download or use, and there are not any licensing fees. Plus, it is freely modified and redistributed, since the language is obtainable under an open-source license - despite the very fact that it's copyright

### **5.3.3Limitations of Python**

Even though, there are numerous advantages, but, this language doesn’t have much scope for ERP packages. It’s difficult to work together with other languages because of declaring variable “types” and cast “values”. It’s also weak in Mobile Computing. As python uses the interpreter in the place of a compiler, the execution of result will be slower.

* Python is a very high-level programming language, yet it is effortless to learn. Anyone can learn to code in Python in just a few hours or a few days. Mastering Python and all its advanced concepts, packages and modules might take some more time. However, learning the basic Python syntax is very easy, as compared to other popular languages like C, C++, and Java.
* Python has an extensive standard library available for anyone to use. This means that [programmers](https://www.simplilearn.com/how-to-become-programmer-article) don’t have to write their code for every single thing unlike other programming languages. There are libraries for image manipulation, databases, unit-testing, expressions and a lot of other functionalities. In addition to the standard library, there is also a growing collection of thousands of components, which are all available in the [Python Package Index](https://pypi.org/).
* When a programming language is interpreted, it means that the source code is executed line by line, and not all at once. Programming languages such as C++ or Java are not interpreted, and hence need to be compiled first to run them. There is no need to compile Python because it is processed at runtime by the interpreter.

## Differ from Other Languages Like PHP or Ruby

Python stands out because it's easy to be told and straightforward to grasp. Many consider Ruby a good place to start out, like Python, yet the latter contains a four-year advantage. This is often because it is easy to crossover between the 2 languages. Both Ruby and Python share a major amount of growth within the job market, so choosing either language would be beneficial in terms of a career. PHP is additionally used often though the applying is different. Ultimately, it comes all the way down to what you may be developing, as each language has its niche.

## Languages Used:

PHP may be a server scripting language that's primarily wont to create dynamic and interactive websites. it's the most effective language for creating HTML content, and may be wont to build anything from a straightforward blog to a large, corporate style website.

Python could be a high-level, object-oriented general-purpose language, that's versatile and may be used for nearly anything. Ruby is a high-level, object-oriented language that's accustomed work with web application and data entities; it exists to require the main target aloof from query tasks. Ruby is most famous for its dynamic type system which performs type checking during runtime. It also features automatic memory management.

TIP: When a language is stated as "high-level" it's because the syntax and commands it recognizes are closer to human language rather than that of a computer. The term high-level was initially accustomed describe languages that aren't locked all the way down to a specific variety of computer.

Out of the three languages, Python is that the best for absolute beginners and is commonly recommended by programmers because it uses a syntax that emphasizes simplicity and simple use. Whereas, Ruby is healthier utilized by programmers that have experience with other languages. PHP, on the opposite hand, is best fitted to developers who are accustomed working with C languages.

## CHAPTER 6

## PROJECT DESCRIPTION

**6.1 System Implementation**

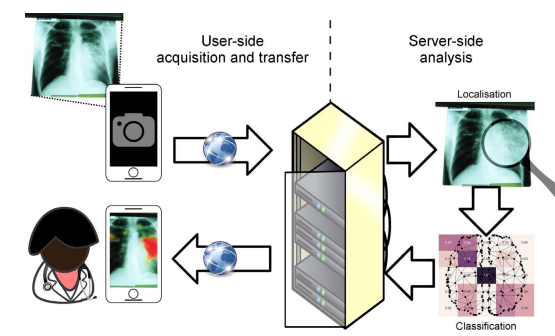
****

Fig no: 6.1 Proposed image processing pipeline.

Possible workflow for a teleradiological TB detection service for health care providers working remotely (schematic). After taking a photograph of the original chest X-ray with a smartphone application, the image is transferred wirelessly to a remote server, where image analysis is carried out in two steps: anomaly detection and classification. The server then sends the image with an overlaid heatmap and a structured report back to the user (health care provider), who may use it to augment his/her clinical judgement for optimal management of the patient.

## 6.1.1 Image Acquisition

Image augmentation For better performance of the neural network, we'd like an outsized training dataset which provides good learning experience to the network. Image augmentation techniques are used to virtually increase the dimensions of training data which helps the neural network classifier to perform well.

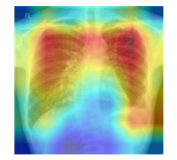


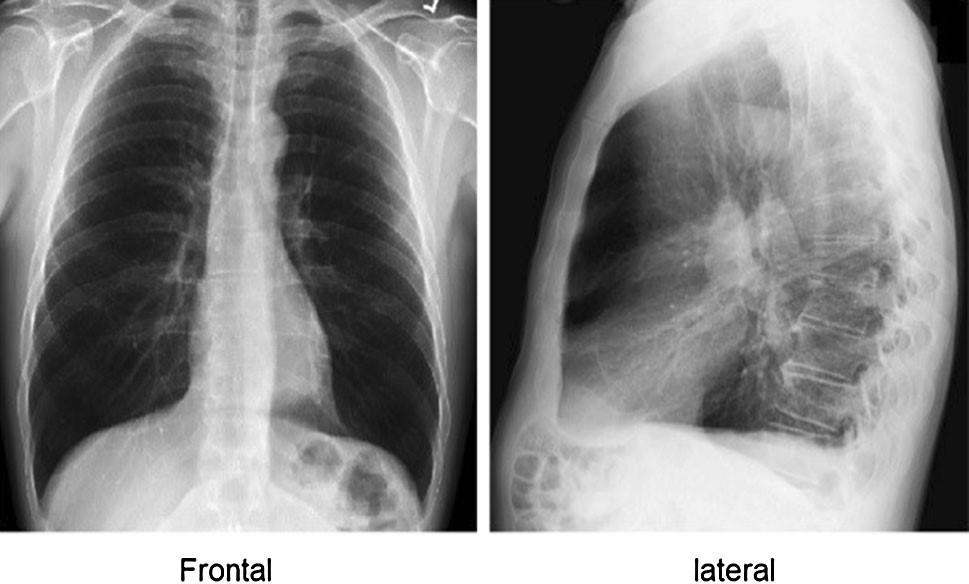
Fig.no.6.1.1 Score-CAM heat map on a chest X-ray image showing that different regions of the image were used in decision making by the CNN.

It creates training images artificially by using alternative ways of combination of multiple processing which are arbitrary rotations, shifts like left shift and right shift, shears and flips.

## 6.2.1 Image pre-processing:

Image processing may be a method want to perform basic operations like extraction and conversion on a picture, to induce an enhanced image by getting the various features and values from it. this can be a type of signal processing within which a picture or set of images is taken as an input then the result we get is also within the sort of image or features related to that image. The three main steps of the digital technique process are pre-processing, enhancement, and restoration. Using the image processing technique, we take the test images from the unhealthy disease (human x-ray) and also together with these we train the images from the dataset so that in the pre-processing stage the features are properly extracted from the photographs. a number of the fundamental steps involved in image processing are Acquisition, Enhancement, Restoration, Compression and Segmentation

Pre-processing images commonly involves removing low-frequency ground noise, normalizing the intensity of the individual particle’s images, removing reflections, and masking portions of images. Image pre-processing is that the technique of enhancing data Furthermore, procedure of image pre-processing involved cropping of all the photographs manually, making the square round the leaves, so as to spotlight the region of interest (human chest x-ray).



## 

Fig .no:6.2.1 Example of X-ray images and corresponding ground truth lung masks from Kaggle dataset

During the phase of collecting the photographs for the dataset, images with smaller resolution and dimension but 500 pixels weren't considered as valid images for the dataset. additionally, only the pictures where the region of interest was in higher resolution were marked as eligible candidates for the dataset. in this way, it had been ensured that images contain all the needed information for feature learning. Many resources may be found by searching across the net, but their relevance is usually unreliable. within the interest of confirming the accuracy of classes within the dataset, initially grouped by a keywords search, medical experts examined chest x-ray images and labeled all the photographs with appropriate disease acronym.

## Because it is thought, it's important to use accurately classified images for the training and validation dataset. Only in this way may an appropriate and reliable detecting model be developed. during this stage, duplicated images that were left after the initial iteration of gathering and grouping images into classes were aloof from the dataset.

## 6.3 Database Augmentation

An image database means storing high quantities of digital images in a particular location. It also means organizing photos so that they can be shared, accessed quickly and easily. during this project we have taken sizable amount of human chest x-ray images into datasets.

## Input Design

The input design is that the link between the data system and therefore the user. It comprises the developing specification and procedures for data preparation and people steps are necessary to place transaction data in to a usable form for processing may be achieved by inspecting the pc to read data from a written or printed document or it can occur by having people keying the info directly into the system. the planning of input focuses on controlling the number of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the method simple. The input is meant in such the way so it provides security and easy use with retaining the privacy.

## Objectives

* + - 1. Input Design is that the process of converting a user-oriented description of the input into a computer-based system. This design is very important to avoid errors within the data input process and show the proper direction to the management for getting correct information from the computerized system.
      2. Achieved by creating user-friendly screens for the information entry to handle large volume of information. The goal of designing input is to create data entry easier and to be free from errors. the information entry screen is meant in such the way that each one the info manipulates are often
      3. When the information is entered it'll check for its validity. Data is entered with the assistance of screens. Appropriate messages are provided as when needed in order that the user won't be in maize of instant. Thus the target of input design is to make an input layout that's easy to follow.

## Output Design

A High-quality output is one, which meets the necessities of the top user and presents the knowledge clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it's determined how the knowledge is to be displaced for immediate need and also the text output. it's the foremost important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to assist user decision-making.

* + - * 1. Designing computer output should proceed in an organized, well thought out manner; the proper output must be developed while ensuring that every output element is meant so people will find the system can use easily and effectively
        2. Select methods for presenting information.
        3. Create document, report, or other formats that contain information produced by the system. The output style of an data system should accomplish one or more of the subsequent objectives.
* Convey information about past activities, current status or projections of the
* Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

## Database Design

The direction System (DBMS) consists of a group of interrelated data and a collection of programs to access that data. the gathering of knowledge usually cited as database. the first key goal of DBMS is to supply an environment that's both convenient and efficient to use in retrieving and storing data information.

## Code Design

A code design may be a document that sets rules for the planning of a brand- new development. it's a tool which will be employed in the planning and planning process, but goes further and is more regulatory than other varieties of guidance commonly utilized in land planning system over recent decades. It is thought of as a process and document – and thus a mechanism – which operationalizes design guidelines or standards which are established through a program process. The program or framework is that the vision. It should be among a design rationale that explains why, followed by a code that offers instructions to the suitable degree or precision which is operational.

## 6.4 CNN Classification

A CNN may be a higher-level language of deep learning. CNN models are trained using large collections of images taken across different sources. From these large collections of databases CNN models can learn rich feature extraction and presentation for a large range of images. CNN is taken into account together of the simplest techniques used for features extraction.

After the segmentation of image into clusters the CNN classification of these clusters happens and we get different images like black and white image, query image and also the segmented image. So once these images are formed the area affected more by the disease can be easily visible and then can be tested properly and accurately. Convolutional networks were inspired by biological processes and are variations of multilayer perceptron designed to use minimal amounts of pre-processing.

They need wide applications in image and video recognition, recommender systems and language processing. Convolutional neural networks (CNNs) contain multiple layers of receptive fields. These are small neuron collections which process portions of the input image. The outputs of those collections are then tiled in order that their input regions overlap, to get a higher-resolution representation of the initial image; this can be repeated for each such layer. Tiling allows CNNs to tolerate translation of the input image.

Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters. They also incorporate various combinations of convolutional and fully connected layers, with point wise nonlinearity applied at the top of or after each layer. A convolution operation on small regions of input is introduced to scale back the number of free parameters and improve generalization. One major advantage of convolutional networks is that the use of shared weight in convolutional layers, which implies that the identical filter (weights bank) is employed for every pixel within the layer; this both reduces memory footprint and improves performance.

Image normalization should be performed before training and evaluating DL models. Reducing the dimension of input images is an initial step in ML. CNN models for classification applications have specific image sizes. Thus, the models first use the input image size and then adjust to the various sizes of convolutional layers; several mathematical formulas and techniques are used in this process.

**6.5 DATASETS AND EXPERIMENTAL SETTINGS**

**Datasets**

[https://ceb.nlm.nih.gov/repositories/tuberculosis-chest-x-rayimage-data-sets/](https://ceb.nlm.nih.gov/repositories/tuberculosis-chest-x-rayimage-data-sets/%20) Shenzhen Hospital X-ray Set: X-ray images in this data set have been collected by Shenzhen No.3 Hospital in Shenzhen, Guangdong province, China. The xrays were acquired as part of the routine care at Shenzhen Hospital. The set contains images in JPEG format. There are 336 normal x-rays and 326 abnormal x-rays showing various manifestations of tuberculosis. Montgomery: X-ray images in this data set have been acquired from the tuberculosis control program of the Department of Health and Human Services of Montgomery County, MD, USA. This set contains 138 posterior-anterior x-rays, of which 80 x-rays are normal and 58 x-rays are abnormal with manifestations of tuberculosis. All images are de-identified and available in DICOM format. The set covers a wide range of abnormalities, including effusions and miliary patterns

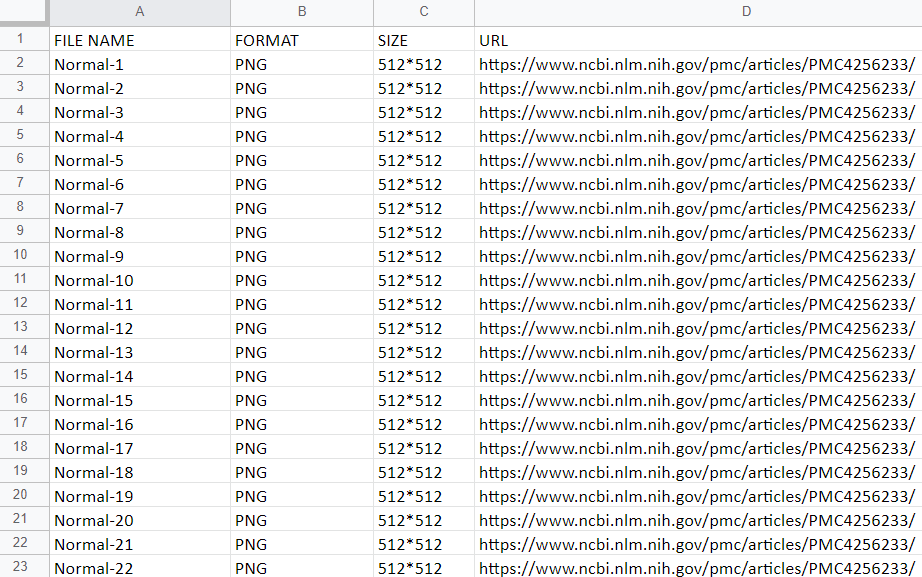


Table No :6.5.1 Gather the normal data in data set

# Table No : 6.5.2 Gather the tuberculosis data in dataset

# SETTINGS

# The software used for the simulation is Matlab. The Image Processing and Neural Network Toolboxes of this product are used. The settings of the simulations are

# All data is shuffled as to never have a repeat sequence of the same data

# A mini batch of 80 is used

# pochs specified as a maximum

# 10% of all data is used for verification

# A fixed random generator for the initial weight selection is used. This ensures that all simulations start with the same random weight values selected.

# Subsequently the results do not deviate. This method is used in all simulations.

# CHAPTER 7

# DATASETS DESCRIPTION

**7.1 DATASETS DESCRIPTION**

In this work, Kaggle Chest X-ray images and corresponding lung mask dataset [69] were used for training the lung segmentation models, where 704 X-ray images and their corresponding ground truth lung masks are available. All mask were annotated by expert radiologists; sample X-ray images and masks are shown in Figure 4. There are 1200 normal X-ray images and 796 abnormal (infected lung) X-ray images available in the dataset. Therefore, U-Net networks were trained with both normal and abnormal images.

Both the Montgomery and Shenzhen datasets contain both active and inactive TB cases. The Belarus dataset contains only active TB cases. Images can be categorized into two types, which are digital radiography (DR) and computed radiography (CR). Both types can be switched using conversion methods through software. In the Belarus dataset, some images (six samples) are negated because they do not contain the ROI, especially the lung or CXR, and were taken in lateral view, which does not allow for detecting TB. The total number of images in the Belarus dataset used in the experiment is 298.

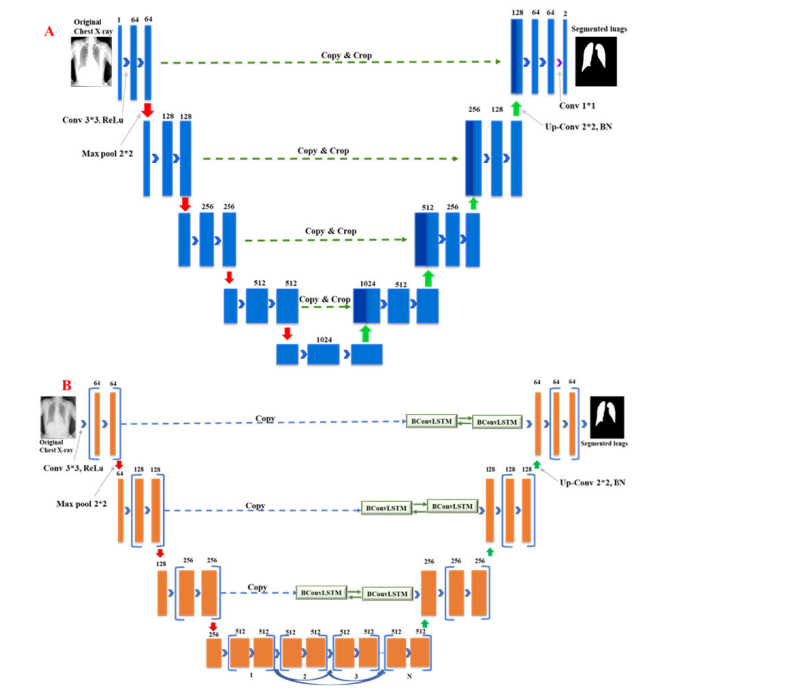
****

Fig no 7.1: Architecture of A) original U-Net and B) modified U-Net.

**7.2 TB CLASSIFICATION**

As mentioned earlier, there are two different experiments (using non-segmented and segmented lungs X-ray images) were conducted for the classification of TB and normal (non-TB) cases. The comparative performance for different CNNs for the binary classification is shown in Table 5. It is apparent from Table 5 that all the evaluated pre-trained models perform very well in classifying TB and normal images in this two-class problem. Among the networks trained with X-ray images without segmentation, CheXNet is performing better for classifying the X-ray images. Even though CheXNet is shallower than DenseNet201, it was originally trained on X-ray images provides it additional benefits in classifying X-ray images and it is showing better performance than DenseNet201. It is not necessary that deeper network will perform better rather CheXNet is a very good example of transfer learning and it outperforms other networks for this problem. Image normalization should be performed before training and evaluating DL models. Reducing the dimension of input images is an initial step in ML. CNN models for classification applications have specific image sizes. Thus, the models first use the input image size and then adjust to the various sizes of convolutional layers; several mathematical formulas and techniques are used in this process. **NLM dataset**: National Library of Medicine (NLM) in U.S. [25] has made two lung X-ray datasets publicly available: the Montgomery and Shenzhen datasets. The Montgomery County (MC) and the Shenzhen, China (CHN) databases are comprised of 138 and 667 posterior-anterior (PA) chest X-ray images respectively. The resolution of the images of MC database was either 4, 020 × 4.892 or 4, 892 × 4.020 pixels whereas that for CHN database was variable but around 3000 × 3000 pixels. In the MC database, out of 138 chest X-ray images, 58 images were taken from different TB patients and 80 images were from normal subjects. In the CHN database, out of 662 chest X-ray images.

**NIAID TB dataset**: NIAID TB portal program dataset [71], which contains about 3000 TB positive CXR images from about 3087 cases. All images were collected from seven different countries and all images are in Portable Network Graphics (PNG) format. In this study, we have used 2800 TB positive CXR images out of 3000 images. 200 poor quality images were discarded from this database.

**RSNA CXR dataset:** RSNA pneumonia detection challenge dataset [72], which is comprised of about 30,000 chest X-ray images, where 10,000 images are normal and others are abnormal and lung opacity images. All images are in Digital Imaging and Communications in Medicine (DICOM) format. To create a normal database of 3,500 chest X-ray images for this study, 3,094 normal images were taken from this database and rest of the 406 normal images were taken from the NLM database. However, the number of TB infected images by combining NLM and Belarus dataset was 700 and from NIAID TB dataset was 2800. In total, there were 3500 TB infected and 3500 normal X-ray images were used in this study.

**Belarus dataset**: Belarus Set [70] was collected for a drug resistance study initiated by the National Institute of Allergy and Infectious Diseases, Ministry of Health, Republic of Belarus. The dataset contains 306 CXR images of 169 patients. Chest radiographs were taken using the Kodak Point-of-Care 260 system and the resolution of the images was 2248 × 2248 pixels. All the images of this database are TB infected images.

**Visualization**

Visualization techniques are used to illustrate important information; the t-SNE visualization technique will confirm that NN layers can create discriminating features between both classes, TB and normal CXR images. The t-SNE technique is better in visualizing high-dimensional data into a two-dimensional map. The t-SNE technique was implemented on a Python platform, with multiple parameters as dimensions and a perplexity-effective number of neighbours The parameters were modified from default values to confirm the performance of all networks on the combined dataset; the t-SNE visualization. From these figures, we see that the TB and no TB samples are clearly separated using both the segmentation and the augmentation . It is essential to observe the efficiency of training an ML model to reveal its network learning distribution in relation to the various available data. Thus, during our experiments, we demonstrate the learning performance of the five CNN models using the combined dataset, which contains a higher number of CXR images, and this is shown using grad-CAM-based heat maps generated for the following original cases:

* Normal CXR images
* Segmented CXR images
* Augmentation in non-segmented CXR images
* Augmentation in segmented CXR images

CNN models for each sample, along with their heat maps on the normal CXR images, segmented CXR images, augmented normal CXR images, and augmented segmented CXR images. From these figures, we see that the presence of TB is more correctly visualized in the segmented and augmented images .

We hope to increase the number of normal no TB CXR cases in the combined dataset to equal the number of abnormal TB CXR cases to improve the system’s generalization ability.

As mentioned earlier, it is important to see where network is learning for the relevant area of the X-ray images or it is learning from anywhere and any non-relevant information for classification.

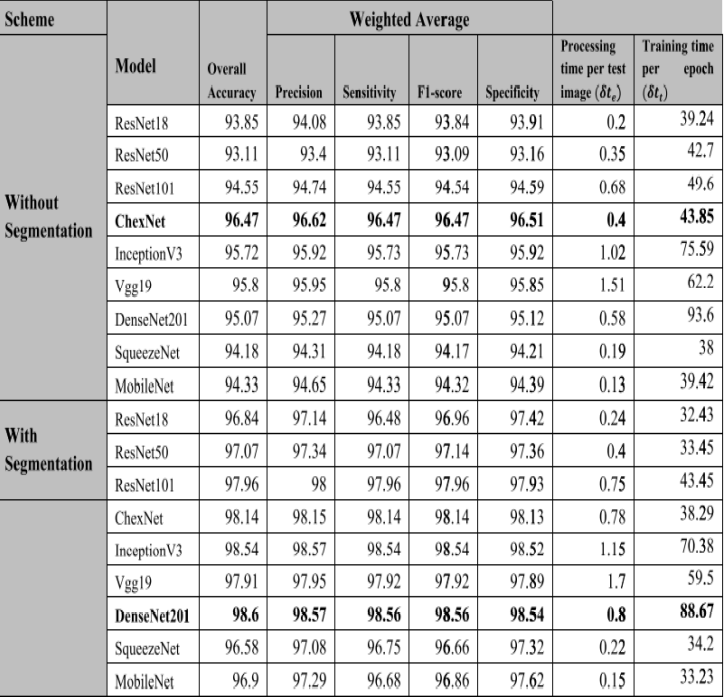


Table 7.2.1: Comparative performance of different models for TB classification using with and without segmented data.

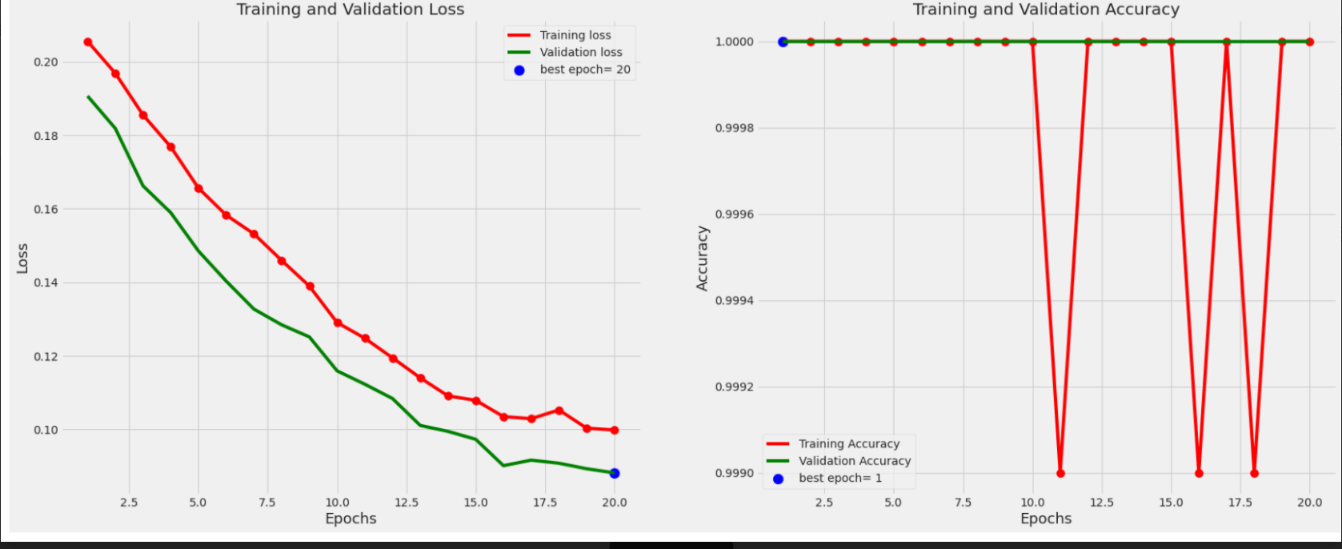
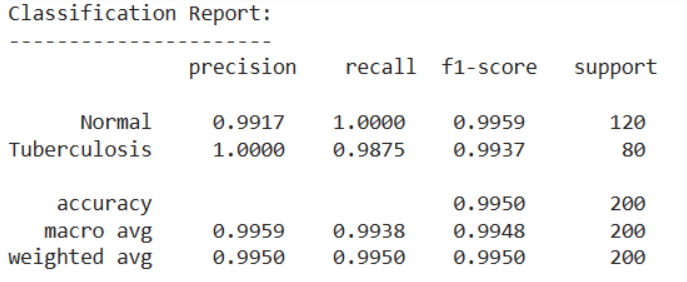
****

Fig no .7.2.1:Comparison of the ROC curves for Normal, and Tuberculosis classification using CNN based models for non-segmented (A) and segmented(B) CXR images.



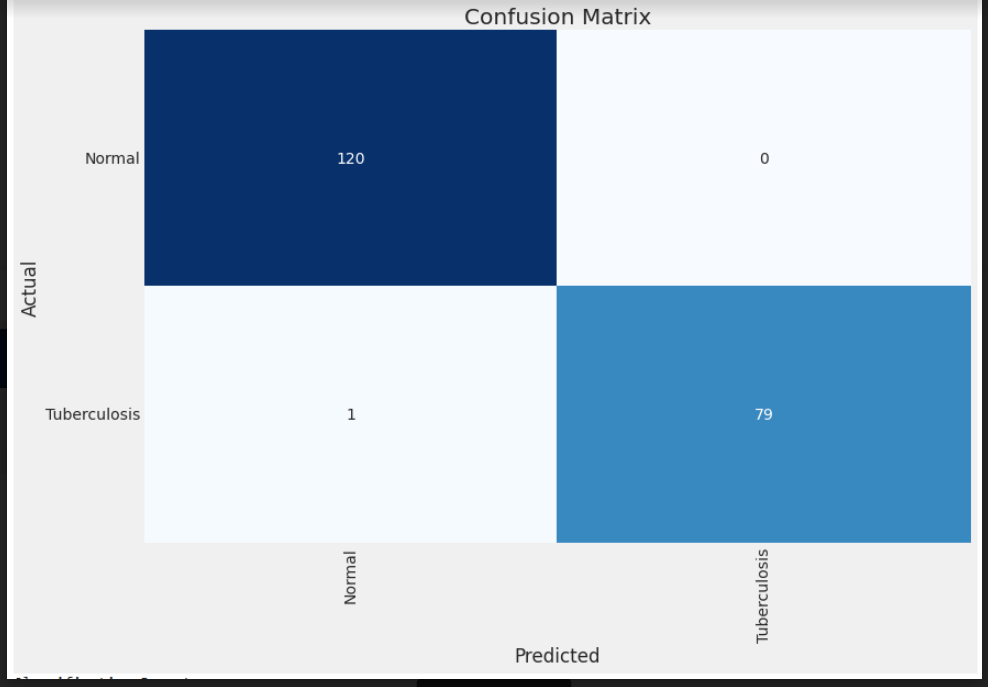


Fig no: 7.2.2 Confusion matrix for Normal and Tuberculosis (TB) classification for ChexNet model without segmented X-ray (A), and DenseNet201 model with segmented X-ray (B).

**SUPPLEMENTARY MATERIALS**

A chest X-ray database of 796 Tuberculosis patients’ image and 1200 normal images were released. This database was created from the 4 publicly available databases, which are referenced in the database. https://www.kaggle.com/ tawsifurrahman/tuberculosis-tb-chest-x ray-dataset

**preprocessing**

The size of the input images for different CNNs were different and therefore the datasets were preprocessed to resize the X-Ray images. In segmentation problem, for original U-Net and modified U-Net, the images were resized to 256 × 256 pixels. In classification problem, for InceptionV3 the images were resized to 227 × 227 pixels whereas for ResNet, DenseNet, ChexNet, VGG, MobileNetV2 and SqueezeNet, the images were resized to 224 × 224 pixels. All images were normalized using Z-score normalization using image database mean and standard deviation.

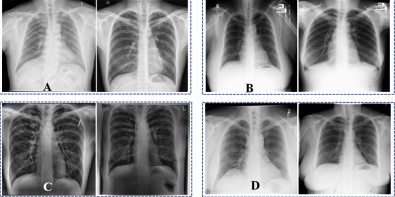
****

Fig no 7 **:**Example of CXR images from different datasets. (A) CHN dataset, (B) MC dataset, (C) Belarus dataset and (D) RSNA dataset.

## CHAPTER 8

## METHODOLOGY

During our experiment, we study and analyze different DL methods for TB detection in a patient’s radiography. We propose an automatic TB detection system for CXR images in the thesis. We study the system by incorporating the above observations. The system uses segmented lung CXR images and recent pre-trained CNN models. It is evaluated on multiple datasets and cross-dataset scenarios. The system includes several blocks, and each block will be described below. Each block depends on the previous one, and the proposed model can be considered a consequential system. Each block in the figure corresponds to one main stage of the proposed system. Finally, the t-SNE technique was implemented on a python platform and dimensions, maximum iterations, perplexity- effective number of neighbors etc. parameters were modified from default values in order to confirm the performance of the best network . To achieve our objectives, the research methodology involves the following steps:

* System design
* Dataset preparation
* Convenient CNN
* Comparison with different systems using several performance metrics

However, we will investigate the following five CNN models because they are comparatively newer and more efficient:

* Xception (2016)
* Inception-ResNet-V2 (2016)
* ResNet-50 (2017)
* MobileNet (2017)
* EfficientNet (2019)

Zip code population data was downloaded from the American Community Survey (ACS) 5 year data and used for calculations of incidence rate. Similarly, income data by zip code was collected from ACS 14, 5 year data. Race/Ethnicity composition for Massachusetts was obtained from the TB dataset and calculated by population data from U.S. Census Bureau.

The TB dataset was linked to the genotypic database from the CDC’s TB Genotyping Information Management System database (TBGIMS) to be able to create cluster data. TB cluster definition: A genotype cluster was defined as two or more TB patients with matching genotypes linked by time and space, when the time period was defined as within a three year period, and space was defined as a genotype cluster in the same geographic area defined by no more than 50 km radius, [performed by using radius around a point (TB case. A total of 85 individuals divided into 32 clusters were identified.

The choropleth maps of TB incidence cases, TB clusters, and TB hot spots gives a spatial understanding of the distribution of TB cases in Massachusetts for targeted interventions.

Once the arrangements with the characteristics of the images for TB and Normal are obtained, the labels are created in a text document to name each one of the images that will be used for the training of the program. Within the processing program, labels and characteristics are called and relationships between labels and characteristics are created to later be converted into arrangements.

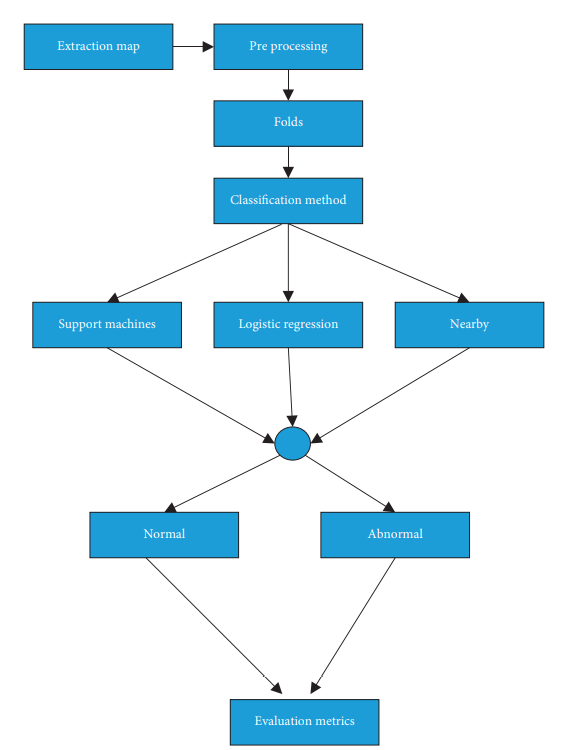


Fig no.8: Overview of the complete system using methodology in tuberculosis detection system

**APPENDIX A**

**SOURCE CODE**

import pandas as pd

import numpy as np

import os

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'

import time

import matplotlib.pyplot as plt

import cv2

import seaborn as sns

sns.set\_style('darkgrid')

import shutil

from sklearn.metrics import confusion\_matrix, classification\_report

from sklearn.model\_selection import train\_test\_split

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.layers import Dense, Activation,Dropout,Conv2D, MaxPooling2D,BatchNormalization

from tensorflow.keras.optimizers import Adam, Adamax

from tensorflow.keras.metrics import categorical\_crossentropy

from tensorflow.keras import regularizers

from tensorflow.keras.models import Model

from tensorflow.keras import backend as K

import time

from tqdm import tqdm

from sklearn.metrics import f1\_score

from IPython.display import YouTubeVideo

import sys

if not sys.warnoptions:

    import warnings

    warnings.simplefilter("ignore")

pd.set\_option('display.max\_columns', None)  # or 1000

pd.set\_option('display.max\_rows', None)  # or 1000

pd.set\_option('display.max\_colwidth', None)  # or 199

print ('Modules loaded')

def print\_in\_color(txt\_msg,fore\_tupple=(0,255,255),back\_tupple=(100,100,100)):

    #prints the text\_msg in the foreground color specified by fore\_tupple with the background specified by back\_tupple

    #text\_msg is the text, fore\_tupple is foregroud color tupple (r,g,b), back\_tupple is background tupple (r,g,b)

    # default parameter print in cyan foreground and gray background

    rf,gf,bf=fore\_tupple

    rb,gb,bb=back\_tupple

    msg='{0}' + txt\_msg

    mat='\33[38;2;' + str(rf) +';' + str(gf) + ';' + str(bf) + ';48;2;' + str(rb) + ';' +str(gb) + ';' + str(bb) +'m'

    print(msg .format(mat), flush=True)

    print('\33[0m', flush=True) # returns default print color to back to black

    return

# example default print

msg='test of default colors'

print\_in\_color(msg)

def make\_dataframes(sdir):

    filepaths=[]

    labels=[]

    classlist=sorted(os.listdir(sdir) )

    for klass in classlist:

        classpath=os.path.join(sdir, klass)

        if os.path.isdir(classpath):

            flist=sorted(os.listdir(classpath))

            desc=f'{klass:25s}'

            for f in tqdm(flist, ncols=130,desc=desc, unit='files', colour='blue'):

                fpath=os.path.join(classpath,f)

                filepaths.append(fpath)

                labels.append(klass)

    Fseries=pd.Series(filepaths, name='filepaths')

    Lseries=pd.Series(labels, name='labels')

    df=pd.concat([Fseries, Lseries], axis=1)

    train\_df, dummy\_df=train\_test\_split(df, train\_size=.8, shuffle=True, random\_state=123, stratify=df['labels'])

    valid\_df, test\_df=train\_test\_split(dummy\_df, train\_size=.5, shuffle=True, random\_state=123, stratify=dummy\_df['labels'])

    classes=sorted(train\_df['labels'].unique())

    class\_count=len(classes)

    sample\_df=train\_df.sample(n=50, replace=False)

    # calculate the average image height and with

    ht=0

    wt=0

    count=0

    for i in range(len(sample\_df)):

        fpath=sample\_df['filepaths'].iloc[i]

        try:

            img=cv2.imread(fpath)

            h=img.shape[0]

            w=img.shape[1]

            wt +=w

            ht +=h

            count +=1

        except:

            pass

    have=int(ht/count)

    wave=int(wt/count)

    aspect\_ratio=have/wave

    print('number of classes in processed dataset= ', class\_count)

    counts=list(train\_df['labels'].value\_counts())

    print(counts[0], type(counts[0]))

    print('the maximum files in any class in train\_df is ', max(counts), '  the minimum files in any class in train\_df is ', min(counts))

    print('train\_df length: ', len(train\_df), '  test\_df length: ', len(test\_df), '  valid\_df length: ', len(valid\_df))

    print('average image height= ', have, '  average image width= ', wave, ' aspect ratio h/w= ', aspect\_ratio)

    return train\_df, test\_df, valid\_df, classes, class\_count

sdir=r'../content/drive/MyDrive/TB\_Chest\_Radiography\_Database'

train\_df, test\_df, valid\_df, classes, class\_count=make\_dataframes(sdir)

def trim(df, max\_samples, min\_samples, column):

    df=df.copy()

    classes=df[column].unique()

    class\_count=len(classes)

    length=len(df)

    print ('dataframe initially is of length ',length, ' with ', class\_count, ' classes')

    groups=df.groupby(column)

    trimmed\_df = pd.DataFrame(columns = df.columns)

    groups=df.groupby(column)

    for label in df[column].unique():

        group=groups.get\_group(label)

        count=len(group)

        if count > max\_samples:

            sampled\_group=group.sample(n=max\_samples, random\_state=123,axis=0)

            trimmed\_df=pd.concat([trimmed\_df, sampled\_group], axis=0)

        else:

            if count>=min\_samples:

                sampled\_group=group

                trimmed\_df=pd.concat([trimmed\_df, sampled\_group], axis=0)

    print('after trimming, the maximum samples in any class is now ',max\_samples, ' and the minimum samples in any class is ', min\_samples)

    classes=trimmed\_df[column].unique()# return this in case some classes have less than min\_samples

    class\_count=len(classes) # return this in case some classes have less than min\_samples

    length=len(trimmed\_df)

    print ('the trimmed dataframe now is of length ',length, ' with ', class\_count, ' classes')

    return trimmed\_df, classes, class\_count

max\_samples=500

min\_samples=500

column='labels'

train\_df, classes, class\_count=trim(train\_df, max\_samples, min\_samples, column)

def balance(df, n, working\_dir, img\_size):

    df=df.copy()

    print('Initial length of dataframe is ', len(df))

    aug\_dir=os.path.join(working\_dir, 'aug')# directory to store augmented images

    if os.path.isdir(aug\_dir):# start with an empty directory

        shutil.rmtree(aug\_dir)

    os.mkdir(aug\_dir)

    for label in df['labels'].unique():

        dir\_path=os.path.join(aug\_dir,label)

        os.mkdir(dir\_path) # make class directories within aug directory

    # create and store the augmented images

    total=0

    gen=ImageDataGenerator(horizontal\_flip=True,  rotation\_range=20, width\_shift\_range=.2,

                                  height\_shift\_range=.2, zoom\_range=.2)

    groups=df.groupby('labels') # group by class

    for label in df['labels'].unique():  # for every class

        group=groups.get\_group(label)  # a dataframe holding only rows with the specified label

        sample\_count=len(group)   # determine how many samples there are in this class

        if sample\_count< n: # if the class has less than target number of images

            aug\_img\_count=0

            delta=n - sample\_count  # number of augmented images to create

            target\_dir=os.path.join(aug\_dir, label)  # define where to write the images

            msg='{0:40s} for class {1:^30s} creating {2:^5s} augmented images'.format(' ', label, str(delta))

            print(msg, '\r', end='') # prints over on the same line

            aug\_gen=gen.flow\_from\_dataframe( group,  x\_col='filepaths', y\_col=None, target\_size=img\_size,

                                            class\_mode=None, batch\_size=1, shuffle=False,

                                            save\_to\_dir=target\_dir, save\_prefix='aug-', color\_mode='rgb',

                                            save\_format='jpg')

            while aug\_img\_count<delta:

                images=next(aug\_gen)

                aug\_img\_count += len(images)

            total +=aug\_img\_count

    print('Total Augmented images created= ', total)

    # create aug\_df and merge with train\_df to create composite training set ndf

    aug\_fpaths=[]

    aug\_labels=[]

    classlist=os.listdir(aug\_dir)

    for klass in classlist:

        classpath=os.path.join(aug\_dir, klass)

        flist=os.listdir(classpath)

        for f in flist:

            fpath=os.path.join(classpath,f)

            aug\_fpaths.append(fpath)

            aug\_labels.append(klass)

    Fseries=pd.Series(aug\_fpaths, name='filepaths')

    Lseries=pd.Series(aug\_labels, name='labels')

    aug\_df=pd.concat([Fseries, Lseries], axis=1)

    df=pd.concat([df,aug\_df], axis=0).reset\_index(drop=True)

    print('Length of augmented dataframe is now ', len(df))

    return df

def make\_gens(batch\_size, train\_df, test\_df, valid\_df, img\_size):

    trgen=ImageDataGenerator()

    t\_and\_v\_gen=ImageDataGenerator()

    msg='{0:70s} for train generator'.format(' ')

    print(msg, '\r', end='') # prints over on the same line

    train\_gen=trgen.flow\_from\_dataframe(train\_df, x\_col='filepaths', y\_col='labels', target\_size=img\_size,

                                        class\_mode='categorical', color\_mode='rgb', shuffle=True,batch\_size=batch\_size)

    msg='{0:70s} for valid generator'.format(' ')

    print(msg, '\r', end='') # prints over on the same line

    valid\_gen=t\_and\_v\_gen.flow\_from\_dataframe(valid\_df, x\_col='filepaths', y\_col='labels', target\_size=img\_size,

                                       class\_mode='categorical', color\_mode='rgb', shuffle=False, batch\_size=batch\_size)

    # for the test\_gen we want to calculate the batch size and test steps such that batch\_size X test\_steps= number of samples in test set

    # this insures that we go through all the sample in the test set exactly once.

    length=len(test\_df)

    test\_batch\_size=sorted([int(length/n) for n in range(1,length+1) if length % n ==0 and length/n<=80],reverse=True)[0]

    test\_steps=int(length/test\_batch\_size)

    msg='{0:70s} for test generator'.format(' ')

    print(msg, '\r', end='') # prints over on the same line

    test\_gen=t\_and\_v\_gen.flow\_from\_dataframe(test\_df, x\_col='filepaths', y\_col='labels', target\_size=img\_size,

                                       class\_mode='categorical', color\_mode='rgb', shuffle=False, batch\_size=test\_batch\_size)

    # from the generator we can get information we will need later

    classes=list(train\_gen.class\_indices.keys())

    class\_indices=list(train\_gen.class\_indices.values())

    class\_count=len(classes)

    labels=test\_gen.labels

    print ( 'test batch size: ' ,test\_batch\_size, '  test steps: ', test\_steps, ' number of classes : ', class\_count)

    return train\_gen, test\_gen, valid\_gen, test\_steps

img\_size=(224,224)

batch\_size =30

train\_gen, test\_gen, valid\_gen, test\_steps =make\_gens(batch\_size, train\_df, test\_df, valid\_df, img\_size)

def show\_image\_samples(gen ):

    t\_dict=gen.class\_indices

    classes=list(t\_dict.keys())

    images,labels=next(gen) # get a sample batch from the generator

    plt.figure(figsize=(25, 25))

    length=len(labels)

    if length<25:   #show maximum of 25 images

        r=length

    else:

        r=25

    for i in range(r):

        plt.subplot(5, 5, i + 1)

        image=images[i] /255

        plt.imshow(image)

        index=np.argmax(labels[i])

        class\_name=classes[index]

        plt.title(class\_name, color='blue', fontsize=18)

        plt.axis('off')

    plt.show()

show\_image\_samples(train\_gen )

def make\_model(img\_size, lr, mod\_num=3):

    img\_shape=(img\_size[0], img\_size[1], 3)

    if mod\_num == 0:

        base\_model=tf.keras.applications.efficientnet.EfficientNetB0(include\_top=False, weights="imagenet",input\_shape=img\_shape, pooling='max')

        msg='Created EfficientNet B0 model'

    elif mod\_num == 3:

        base\_model=tf.keras.applications.efficientnet.EfficientNetB3(include\_top=False, weights="imagenet",input\_shape=img\_shape, pooling='max')

        msg='Created EfficientNet B3 model'

    elif mod\_num == 5:

        base\_model=tf.keras.applications.efficientnet.EfficientNetB5(include\_top=False, weights="imagenet",input\_shape=img\_shape, pooling='max')

        msg='Created EfficientNet B5 model'

    else:

        base\_model=tf.keras.applications.efficientnet.EfficientNetB7(include\_top=False, weights="imagenet",input\_shape=img\_shape, pooling='max')

        msg='Created EfficientNet B7 model'

    base\_model.trainable=True

    x=base\_model.output

    x=BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001 )(x)

    x = Dense(256, kernel\_regularizer = regularizers.l2(l = 0.016),activity\_regularizer=regularizers.l1(0.006),

                    bias\_regularizer=regularizers.l1(0.006) ,activation='relu')(x)

    x=Dropout(rate=.4, seed=123)(x)

    output=Dense(class\_count, activation='softmax')(x)

    model=Model(inputs=base\_model.input, outputs=output)

    model.compile(Adamax(learning\_rate=lr), loss='categorical\_crossentropy', metrics=['accuracy'])

    msg=msg + f' with initial learning rate set to {lr}'

    print\_in\_color(msg)

    return model

lr=.001

model=make\_model(img\_size, lr) # using B3 model by default

lass LR\_ASK(keras.callbacks.Callback):

    def \_\_init\_\_ (self, model, epochs,  ask\_epoch, dwell=True, factor=.4): # initialization of the callback

        super(LR\_ASK, self).\_\_init\_\_()

        self.model=model

        self.ask\_epoch=ask\_epoch

        self.epochs=epochs

        self.ask=True # if True query the user on a specified epoch

        self.lowest\_vloss=np.inf

        self.lowest\_aloss=np.inf

        self.best\_weights=self.model.get\_weights() # set best weights to model's initial weights

        self.best\_epoch=1

        self.plist=[]

        self.alist=[]

        self.dwell= dwell

        self.factor=factor

    def get\_list(self): # define a function to return the list of % validation change

        return self.plist, self.alist

    def on\_train\_begin(self, logs=None): # this runs on the beginning of training

        if self.ask\_epoch == 0:

            print('you set ask\_epoch = 0, ask\_epoch will be set to 1', flush=True)

            self.ask\_epoch=1

        if self.ask\_epoch >= self.epochs: # you are running for epochs but ask\_epoch>epochs

            print('ask\_epoch >= epochs, will train for ', epochs, ' epochs', flush=True)

            self.ask=False # do not query the user

        if self.epochs == 1:

            self.ask=False # running only for 1 epoch so do not query user

        else:

            msg =f'Training will proceed until epoch {ask\_epoch} then you will be asked to'

            print\_in\_color(msg )

            msg='enter H to halt training or enter an integer for how many more epochs to run then be asked again'

            print\_in\_color(msg)

            if self.dwell:

                msg='learning rate will be automatically adjusted during training'

                print\_in\_color(msg, (0,255,0))

        self.start\_time= time.time() # set the time at which training started

    def on\_train\_end(self, logs=None):   # runs at the end of training

        msg=f'loading model with weights from epoch {self.best\_epoch}'

        print\_in\_color(msg, (0,255,255))

        self.model.set\_weights(self.best\_weights) # set the weights of the model to the best weights

        tr\_duration=time.time() - self.start\_time   # determine how long the training cycle lasted

        hours = tr\_duration // 3600

        minutes = (tr\_duration - (hours \* 3600)) // 60

        seconds = tr\_duration - ((hours \* 3600) + (minutes \* 60))

        msg = f'training elapsed time was {str(hours)} hours, {minutes:4.1f} minutes, {seconds:4.2f} seconds)'

        print\_in\_color (msg) # print out training duration time

    def on\_epoch\_end(self, epoch, logs=None):  # method runs on the end of each epoch

        vloss=logs.get('val\_loss')  # get the validation loss for this epoch

        aloss=logs.get('loss')

        if epoch >0:

            deltav = self.lowest\_vloss- vloss

            pimprov=(deltav/self.lowest\_vloss) \* 100

            self.plist.append(pimprov)

            deltaa=self.lowest\_aloss-aloss

            aimprov=(deltaa/self.lowest\_aloss) \* 100

            self.alist.append(aimprov)

        else:

            pimprov=0.0

            aimprov=0.0

        if vloss< self.lowest\_vloss:

            self.lowest\_vloss=vloss

            self.best\_weights=self.model.get\_weights() # set best weights to model's initial weights

            self.best\_epoch=epoch + 1

            msg=f'\n validation loss of {vloss:7.4f} is {pimprov:7.4f} % below lowest loss, saving weights from epoch {str(epoch + 1):3s} as best weights'

            print\_in\_color(msg, (0,255,0)) # green foreground

        else: # validation loss increased

            pimprov=abs(pimprov)

            msg=f'\n validation loss of {vloss:7.4f} is {pimprov:7.4f} % above lowest loss of {self.lowest\_vloss:7.4f} keeping weights from epoch {str(self.best\_epoch)} as best weights'

            print\_in\_color(msg, (255,255,0)) # yellow foreground

            if self.dwell: # if dwell is True when the validation loss increases the learning rate is automatically reduced and model weights are set to best weights

                lr=float(tf.keras.backend.get\_value(self.model.optimizer.lr)) # get the current learning rate

                new\_lr=lr \* self.factor

                msg=f'learning rate was automatically adjusted from {lr:8.6f} to {new\_lr:8.6f}, model weights set to best weights'

                print\_in\_color(msg) # cyan foreground

                tf.keras.backend.set\_value(self.model.optimizer.lr, new\_lr) # set the learning rate in the optimizer

                self.model.set\_weights(self.best\_weights) # set the weights of the model to the best weights

        if aloss< self.lowest\_aloss:

            self.lowest\_aloss=aloss

        if self.ask: # are the conditions right to query the user?

            if epoch + 1 ==self.ask\_epoch: # is this epoch the one for quering the user?

                msg='press enter to continue or enter a comment  below '

                print\_in\_color(msg)

                comment=input(' ')

                if comment !='':

                    print\_in\_color(comment, (155,245,66))

                msg='\n Enter H to end training or  an integer for the number of additional epochs to run then ask again'

                print\_in\_color(msg) # cyan foreground

                ans=input()

                if ans == 'H' or ans =='h' or ans == '0': # quit training for these conditions

                    msg=f'you entered {ans},  Training halted on epoch {epoch+1} due to user input\n'

                    print\_in\_color(msg)

                    self.model.stop\_training = True # halt training

                else: # user wants to continue training

                    self.ask\_epoch += int(ans)

                    if self.ask\_epoch > self.epochs:

                        print('\nYou specified maximum epochs of as ', self.epochs, ' cannot train for ', self.ask\_epoch, flush =True)

                    else:

                        msg=f'you entered {ans} Training will continue to epoch {self.ask\_epoch}'

                        print\_in\_color(msg) # cyan foreground

                        if self.dwell==False:

                            lr=float(tf.keras.backend.get\_value(self.model.optimizer.lr)) # get the current learning rate

                            msg=f'current LR is  {lr:8.6f}  hit enter to keep  this LR or enter a new LR'

                            print\_in\_color(msg) # cyan foreground

                            ans=input(' ')

                            if ans =='':

                                msg=f'keeping current LR of {lr:7.5f}'

                                print\_in\_color(msg) # cyan foreground

                            else:

                                new\_lr=float(ans)

                                tf.keras.backend.set\_value(self.model.optimizer.lr, new\_lr) # set the learning rate in the optimizer

                                msg=f' changing LR to {ans}'

                                print\_in\_color(msg) # cyan foreground

epochs=40

ask\_epoch=12

ask=LR\_ASK(model, epochs,  ask\_epoch)

callbacks=[ask]

history=model.fit(x=train\_gen,  epochs=epochs, verbose=1, callbacks=callbacks,  validation\_data=valid\_gen,

                 shuffle=False,  initial\_epoch=0)

def tr\_plot(tr\_data, start\_epoch):

    #Plot the training and validation data

    tacc=tr\_data.history['accuracy']

    tloss=tr\_data.history['loss']

    vacc=tr\_data.history['val\_accuracy']

    vloss=tr\_data.history['val\_loss']

    Epoch\_count=len(tacc)+ start\_epoch

    Epochs=[]

    for i in range (start\_epoch ,Epoch\_count):

        Epochs.append(i+1)

    index\_loss=np.argmin(vloss)#  this is the epoch with the lowest validation loss

    val\_lowest=vloss[index\_loss]

    index\_acc=np.argmax(vacc)

    acc\_highest=vacc[index\_acc]

    plt.style.use('fivethirtyeight')

    sc\_label='best epoch= '+ str(index\_loss+1 +start\_epoch)

    vc\_label='best epoch= '+ str(index\_acc + 1+ start\_epoch)

    fig,axes=plt.subplots(nrows=1, ncols=2, figsize=(25,10))

    axes[0].plot(Epochs,tloss, 'r', label='Training loss')

    axes[0].plot(Epochs,vloss,'g',label='Validation loss' )

    axes[0].scatter(index\_loss+1 +start\_epoch,val\_lowest, s=150, c= 'blue', label=sc\_label)

    axes[0].scatter(Epochs, tloss, s=100, c='red')

    axes[0].set\_title('Training and Validation Loss')

    axes[0].set\_xlabel('Epochs', fontsize=18)

    axes[0].set\_ylabel('Loss', fontsize=18)

    axes[0].legend()

    axes[1].plot (Epochs,tacc,'r',label= 'Training Accuracy')

    axes[1].scatter(Epochs, tacc, s=100, c='red')

    axes[1].plot (Epochs,vacc,'g',label= 'Validation Accuracy')

    axes[1].scatter(index\_acc+1 +start\_epoch,acc\_highest, s=150, c= 'blue', label=vc\_label)

    axes[1].set\_title('Training and Validation Accuracy')

    axes[1].set\_xlabel('Epochs', fontsize=18)

    axes[1].set\_ylabel('Accuracy', fontsize=18)

    axes[1].legend()

    plt.tight\_layout

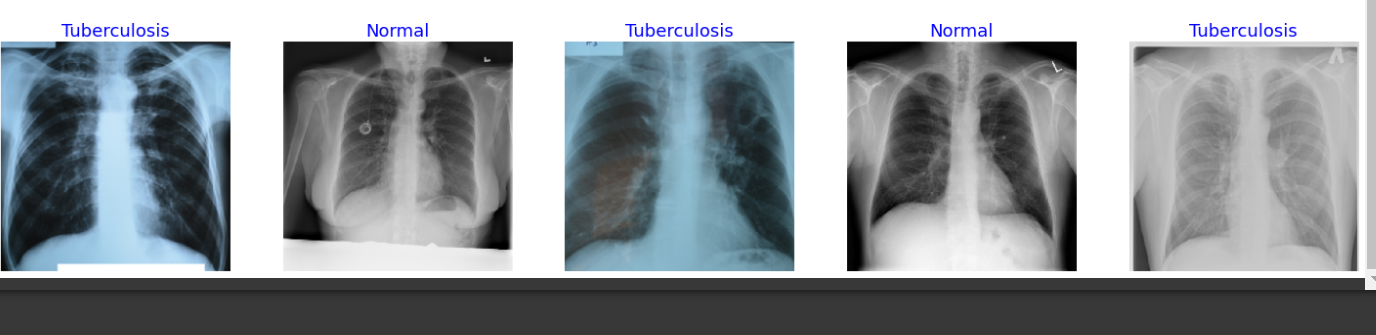
    plt.show()

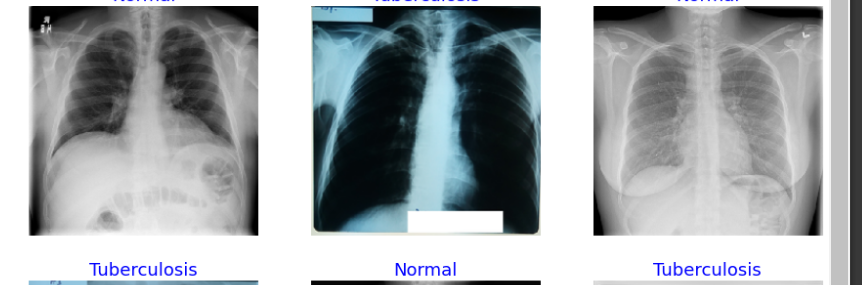
    return index\_loss

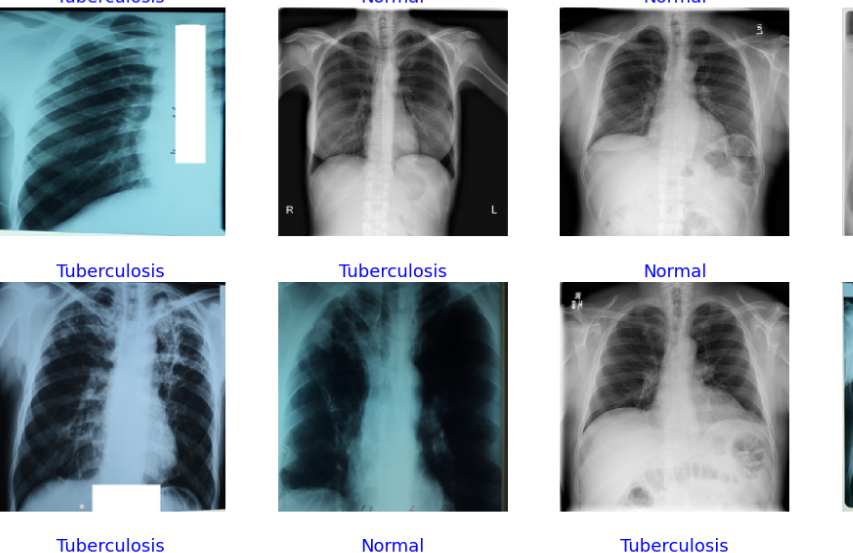
**APPENDIX B**

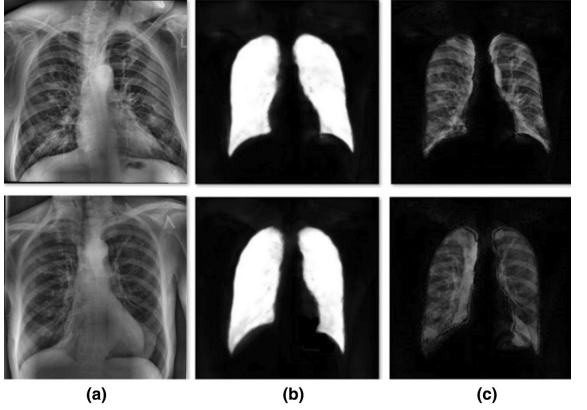
# Screen Shots

## Image table in server where the images are uploaded





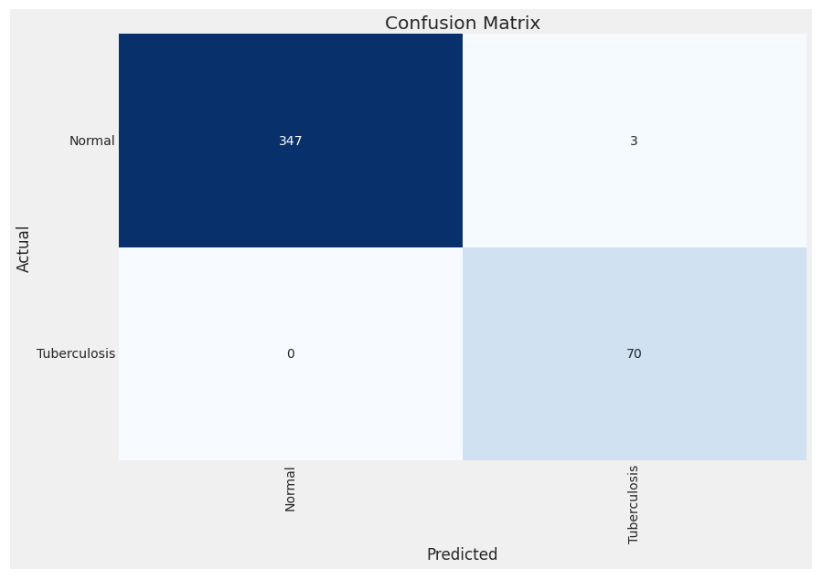


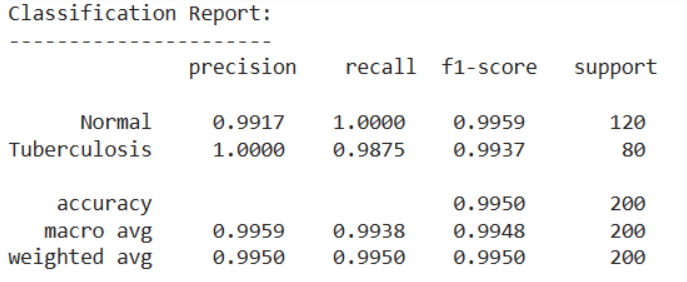


# U-Net output segmentation results for two sample lung CXR images. a shows the original images. b shows the results of U-Net. c show the results after ROI extraction

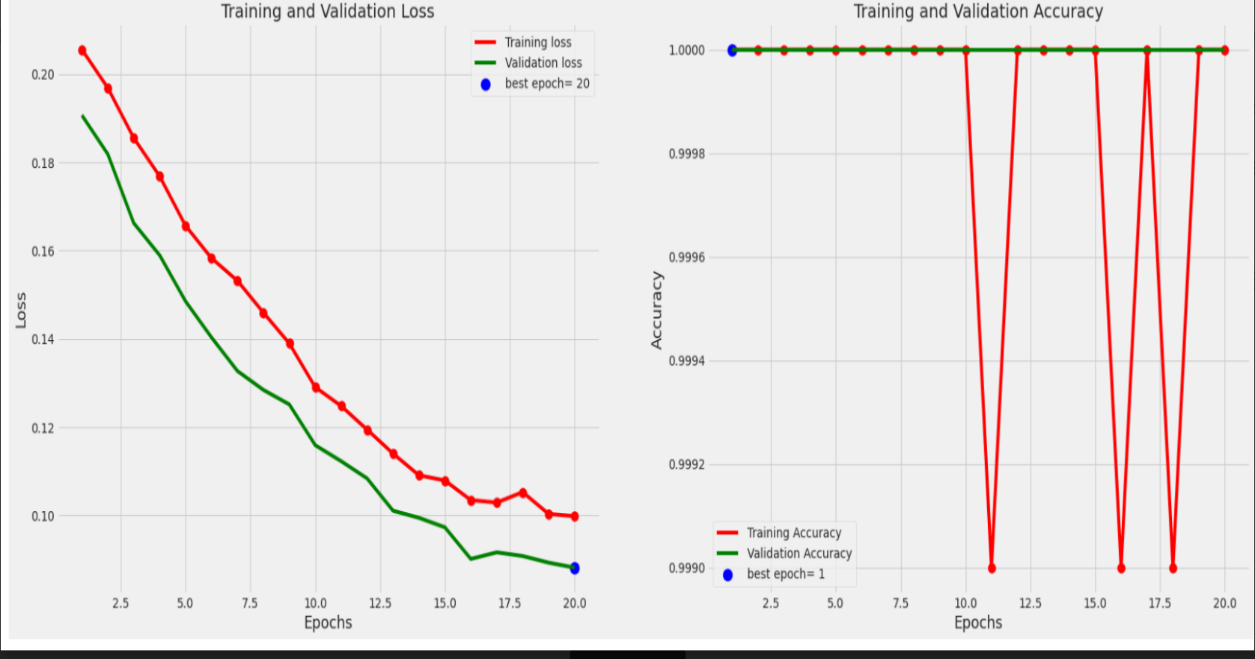
**Clustering Of The Input Image**

## Matlab Command Prompt Where The Affected Area Is Displayed





Help Dialog Displaying the Tuberculosis disease name

****

## CHAPTER 9

**CONCLUSION & FUTURE ENHANCEMENTS**

The results yield solid evidence that preprocessing of images shows a higher accuracy than no preprocessing. Subsequently extracting only the ROI from these images yields even better results with a maximum accuracy of 92.54% the main advantage of the hybrid method is significantly better accuracy by reducing overfitting. This work presents a transfer learning approach with deep Convolutional Neural Networks for the automatic detection of tuberculosis from the chest radiographs. The performance of nine different CNN models were evaluated for the classification of TB and normal CXR image. He classification accuracy, precision and recall for the detection of TB were found to be 96.47%, 96.62%, and 96.47% without segmentation and 98.6%, 98.57%, and 98.56% with segmentation respectively. We compute a set of shape, edge, and texture features as input to a binary classifier, which then classifies the given input image into either normal or abnormal. In this paper, we compare two different established feature sets: one set typically used for object recognition and the other used in image retrieval applications. In future experiments, we will evaluate our system on larger datasets that we will collect using our portable scanners in Kenya. This state-of-the-art performance of our proposed system was compared with the recently published efforts toward the same goal.. Some of these efforts used publicly available datasets, such as [9, 12], whereas others used private datasets, such as [8, 11]. Figure 9 shows the results of U-Net samples and steps for segmenting lung images. Table 4 provides the main information of the experiments using the U-Net. Figure 10 shows the training and validation loss versus epochs for the five CNN models for augmented segmented lung CXR images. From this figure, we see that for ResNet50 and EfficientNetB3, the loss graphs are smooth, and the system are nicely convergine

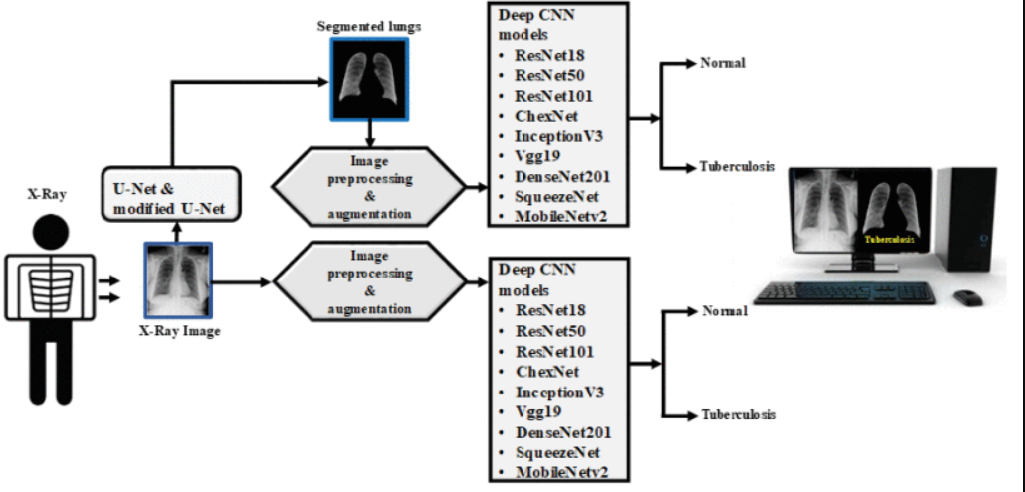


Fig no.9: Overview of the complete system in detection system

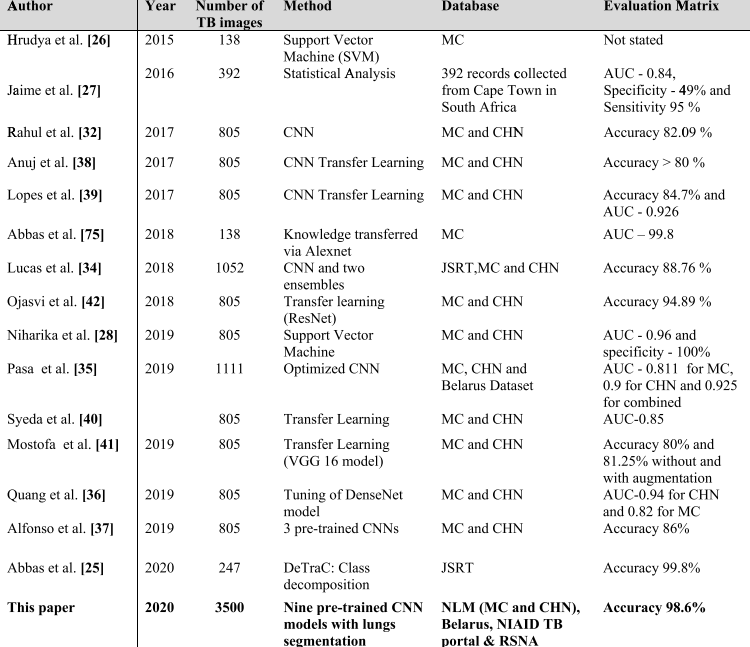


Table no 9: comparison of findings of this study with other recent similar works.

First case of TB miss-classified image was very similar to normal image and radiologist was classified them as normal and therefore, it may be very early stage of TB as it was labelled as TB infected image in the original dataset. However, second to fourth TB infected images are from mild to moderate TB patients and CNN miss-classified them as it is learning from wrong area of the lungs. Therefore, if the segmented lungs can be further segmented into patches which can be used as input to CNN model, which might further enhance the performance. This is the future direction of investigation of this work

**REFERENCE**

1. Bengio Y. Learning Deep architectures for AI. In: Jordan M, ed. Foundations and trendsw in machine learning. Delft, The Netherlands: Now Publishers Inc, 2009
2. B. P. Health. (2020). Belarus Tuberculosis Portal. Accessed: Jun. 9, 2020. [Online]. Available: http://tuberculosis.by/
3. E. Alpaydin, Introduction to Machine Learning, The MIT Press, Cambridge, MA, USA, 2 edition, 2010
4. E. Dzaferovi ˇ c, A. Sokol, A. A. Almisreb, and S. M. Norzeli, ´ “DoS and DDoS vulnerability of IoT: a review,” Sustainable Engineering and Innovation
5. Hinton G E. To recognize shapes, first learn to generate images. Prog Brain Res 2007; 165: 535–547
6. J. L. D ´ıaz-Huerta, A. D. C. Tellez-Anguiano, M. Fraga- ´ Aguilar, J. A. Guti´errez-Gnecchi, and S. Arellano-Calder´on, “Image processing for AFB segmentation in bacilloscopies of pulmonary tuberculosis diagnosis,” PLoS One, vol. 14, no. 7, þ, Article ID e0218861, 2019
7. kaggle. RSNA Pneumonia Detection Challenge [Online]. Available: https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data. [Accessed on 09-June-2020]
8. kaggle. RSNA Pneumonia Detection Challenge. Accessed: Jun. 9, 2020. [Online]. Available: https://www.kaggle.com/c/rsna-pneumoniadetection-challenge/data
9. L. T. Maria Antony and A. Abdullah Hamad, “A theoretical implementation for a proposed hyper-complex chaotic system,” Journal of Intelligent and Fuzzy Systems, vol. 38, no. 3, pp. 2585–2590, 2020, þ
10. Muhammad G, Hossain MS (2021) COVID-19 and Non-COVID19 classification using multi-layers fusion from lung ultrasound images. Information Fusion 72:80–88
11. O. Yadav, K. Passi, and C. K. Jain, ‘‘Using deep learning to classify X-ray images of potential tuberculosis patients,’’ in Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM), Dec. 2018, pp. 2368–2375.
12. Rahman T et al (2020) Reliable Tuberculosis detection using chest X-Ray with deep learning, segmentation and visualization. IEEE Access 8:191586–191601
13. S. Jaeger, S. Candemir, S. Antani, Y.-X. J. Wáng, P.-X. Lu, and G. Thoma, "Two public chest X-ray datasets for computer-aided screening of pulmonary diseases," Quantitative imaging in medicine and surgery
14. Serre T, Kreiman G, Kouh M, Cadieu C, Knoblich U, Poggio T. A quantitative theory of immediate visual recognition. In: Cisek P, Drew T, Kalaska J, eds. Computational neuroscience: theoretical insights into brain function. Progress in Brain Research. Vol 165. Amsterdam, The Netherlands: Elsevier, 2007: pp 33–56.
15. S. Oruc and S. Yeralan, “A meta-study on future work in information and communication technologies,” Heritage and Sustainable Development, vol. 2, no. 2, pp. pp114–122, 2020.
16. Tawsifur Rahman, Amith Khandakar, Muhammad A. Kadir, Khandaker R. Islam, Khandaker F. Islam, Zaid B. Mahbub, Mohamed Arselene Ayari, Muhammad E. H. Chowdhury. (2020) "Reliable Tuberculosis Detection using Chest X-ray with Deep Learning, Segmentation and Visualization". IEEE Access, Vol. 8, pp 191586 - 191601. DOI. 10.1109/ACCESS.2020.3031384.
17. Tiemersma E W, van der Werf M J, Borgdorff M W, Williams B G, Nagelkerke N J D. Natural history of tuberculosis: duration and fatality of untreated pulmonary tuberculosis in HIV negative patients: a systematic review. PLOS ONE 2011; 6: e17601.
18. World Health Organization. Global tuberculosis report, 2016. WHO/HTM/TB/2016.13. Geneva, Switzerland: WHO, 2016
19. Yadav O, Passi K, Jain CK, (2018) Using deep learning to classify X-ray images of potential tuberculosis patients. In: 2018 IEEE international conference on bioinformatics and biomedicine (BIBM), Madrid, Spain, pp. 2368–2375.
20. Y. Kurmi, V. Chaurasia, A. Goel, D. Joshi, and N. Kapoor, “Tuberculosis bacteria analysis in acid fast stained images of sputum smear,” Signal, Image and Video Processing, vol. 15, no. 1, pp. 175–183, 20.

.