Dedication

This paper is dedicated to our beloved parents who have always been a source of inspiration, encouragement and stamina to undertake my higher studies and to face the eventualities of life with zeal, enthusiasm and fear of God.

Acknowledgements

We would like to thank our supervisor (Dr. Sajid Mehmood) who helped and guided us in the perfect way to put our efforts in the right way and give us the chance to turn our thinking into reality. This level of work can't be satisfactorily completed without our knowledgeable supervisor.

We would special thanks to our collaboration partner Shenzhen-Hospital from china to provide us the data for Tuberculosis. It really helped us a lot to further move on the project.

We are also very thankful to Inês Domingues for providing us the breast cancer data. The INbreast database is a mammographic database, with images acquired at a Breast Centre, located in a University Hospital (Hospital de São João, Breast Centre, Porto, Portugal). INbreast has a total of 115 cases (410 images) of which 90 cases are from women with both breasts (4 images per case) and 25 cases are from mastectomy patients (2 images per case). Several types of lesions (masses, calcifications, asymmetries, and distortions) are included. Accurate contours made by specialists are also provided in XML format.

ABSTRACT

Many patients die every year due to lack of treatment or error in diagnosis of a certain disease. Developing an artificial intelligence based detection system can help in early diagnosis and treatment of many diseases.

In this paper, we present a potential approach to having a collection of disease detection mechanisms using deep learning which utilize medical imaging for classification. To demonstrate this experiment we use two of the most major diseases detectable using medical imaging: Tuberculosis (TB) and Breast Cancer.

We have used CNN architecture and compared the performance of the network using three different activation functions. In the case of Breast Cancer, we used the INbreast dataset which has a total of 115 cases (410 images) of which 90 cases are from women with both breasts (4 images per case) and 25 cases are from mastectomy patients (2 images per case). Our network achieves an accuracy of 94.51% in predicting the cancer in breast. In the case of TB, we used the Shenzhen dataset which has 326 normal x-rays and 336 abnormal x-rays showing various manifestations of tuberculosis. Our network achieves an accuracy of 96% in predicting TB in chest X-rays.

To further demonstrate our experiment we have created an API (Application Programming Interface) for both of our detection programs which uses the Flask framework on Python. Using an API we can utilize these detection programs in any OS (operating system), program or device which can be connected to the internet. For public consumption we've created a simple front-end web and mobile application based on PHP and Swift languages, respectively.

Table of Contents

Definitions and Acronyms

IRMI: Intelligent Remote Medical Imaging

TF: Tensor Flow

AI: Artificial Intelligence

TB: Tuberculosis

CNN: Convolution Neural Network

CLAHE: Contrast Limited Adaptive Histogram Equalization

PNG: Portable Network Graphics

List of Figures

List of Tables

1 INTRODUCTION

1.1 Problem Overview

The expeditious advancement of machine learning and especially deep learning perpetuates to fuel the medical imaging community's interest in applying these techniques to amend the precision of cancer screening. Breast cancer is the second leading cause of cancer deaths among women.

Tuberculosis (TB) is a fast spread disease whose symptoms are also show late, which destroy the human lungs badly, and it is responsible for death of more than 1.4 million deaths every year, and it is ranking above HIV/AIDS. Timely diagnosis and treatment is key to full patient recuperation. And it is believed that around 5–10% of the people develop active TB during their life.

Some parts of the body can be affected, such as kidneys, bones, lymph nodes, joints, etc. It can affect anyone. It permeates the air when someone with untreated TB disease sneezes, laughs, coughs or even sings. The World Health Organization estimates that 1.8 billion people proximate to one quarter of the world's population are infected with Mycobacterium tuberculosis (M.tb), the bacteria that causes TB. Last year, 10 million fell ill from TB and 1.6 million died. TB is an airborne disease that can be spread by coughing or sneezing and is the leading cause of infectious disease ecumenical.
Breast cancer is a disease in which cells in the breast grow out of control. There are different kinds of breast cancer. The kind of breast cancer depends on which cells in the breast turn into cancer. A vigorous family history of breast cancer or inherited transmutations in your BRCA1 and BRCA2 genes, you may have a high risk of getting breast cancer.
A mammogram is an X-ray picture of the breast. Medicos utilize a mammogram to probe for early designations of breast cancer. Customary mammograms are the best tests medicos have to find breast cancer early, sometimes up to three years afore it can be felt. It is estimated that ecumenical over 508 000 women died in 2011 due to breast cancer (Ecumenical Health Estimates, WHO 2013). Albeit breast cancer is thought to be a disease of the developed world, virtually 50% of breast cancer cases and 58% of deaths occur in less developed countries (GLOBOCAN 2008).

The expanding and diversifying role of more incipient imaging techniques in breast cancer and tuberculosis has provided radiologists with incrementing datasets and varied multimodality diagnostic implements, all of which are differentially applied in sundry clinical scenarios. Artificial astuteness (AI) offers the opportunity to streamline and integrate the diagnostic expertise of the radiologist, including the apperception and stratification of involute patterns in images.

Our research-based project incorporated into this health care application employs a deep learning approach. Datasets for research purposes are collected from the internet and other sources. We developed a model which is capable of handling multiple diagnostic image testing and is not limited to a few. Based on the performance of the model in terms of accuracy we determine the validity of it.

1.2 Research Questions

Mammography

Input:

Mammography (low-energy X-rays usually around 30 kVp)

Output:

1. **Not a valid digital Mammography**
2. **Normal (-ve)**
3. **Abnormal (+ve)**
    - **3.1 Cancerous (+ve)**
        - 3.1.1 Cysts
        - 3.1.2 Microcysts
        - 3.1.3 Macrocysts
    - **3.2 Calcifications**
        - 3.2.1 Macrocalcifications
        - 3.2.2 Microcalcifications
    - **3.3 Fibroadenomas**
        - 3.3.1 Simple fibroadenomas
        - 3.3.2 Complex fibroadenomas
    - **3.4 Scar tissue**
        - 3.4.1 After lumpectomy
        - 3.4.2 After mastectomy
        - 3.4.3 After breast reconstruction
    - **3.5 Ductal carcinoma in situ (DCIS)**
    - **3.6 Hamartoma**
    - **3.7 Hematoma**
    - **3.8 Invasive ductal carcinoma (IDC)**
    - **3.9 Invasive lobular carcinoma (ILC)**
    - **3.10 Lobular carcinoma in situ (LCIS)**
    - **3.11 Lipomas**
    - **3.12 Lymph nodes**
    - **3.13 Papilloma**

Chest X-ray

Input:

Digital X-ray.

Output:

1. **Not a valid digital chest X-ray**
2. **Normal(-ve)**
3. **Abnormal(+ve)**
    - **3.1 Pneumonia(+ve)**
        - 3.1.1 Bacterial Pneumonia
        - 3.1.2 Viral Pneumonia
        - 3.1.3 Mycoplasma pneumonia
        - 3.1.4 Other pneumonias
    - **3.2 Tuberculosis(+ve)**
        - 3.2.1 Miliary TB
        - 3.2.2 Active TB Disease
    - **3.3 Heart failure and other heart problems**
    - **3.4 Emphysema**
    - **3.5 Lung cancer**
    - **3.6 Positioning of medical devices**
    - **3.7 Fluid or air collection around the lungs**

1.3 Research Objectives

Our goal is to reach higher accuracy in tuberculosis and breast cancer detection than before by using better and improved algorithms along with bigger data sets.

We have researched the Pakistani market and realized the fact that such a service does not exist in the market, or even if it exists it is not of that caliber to be trusted by the vast majority of the public. Technology has enabled millions of Pakistanis to connect with each other. And we believe this technology can be used to connect these people with such a service to improve their lives and make diagnostic testing easier and affordable for everyone. The world is swiftly moving towards automation as it advances technologically.

1.4 Scope

The Automated disease detection in diagnostic imaging tests makes use of image tests such as X-ray and mammogram which create visual representations of a body's interior for disease detection. A deep learning approach is used for accurate detection of diseases in image sets. We'll be implementing this research-based tool on breast cancer and Tuberculosis detections while improving the accuracy. Our automated disease detection system first detects if it is an accurate X-ray or mammogram if so, whether it is normal or an abnormal one. If it's a normal X-ray or mammogram the system doesn't have to go any further, but if it is an abnormal X-ray or mammogram then the system looks for sub categories of the related disease. Also, another featurette would be accessibility from anywhere in the world where there are internet services.

Another goal of this research is to not only to improve the accuracy of previously established research but also on effective ways to integrate all these detection mechanisms into one single AI system for the benefit of humanity. An AI system which boasts the capabilities of handling multiple disease detection becomes a perfect application as an automated health care guide.

1.5 Methodology

Data Collection Methods:

We collected datasets from some pre-established data repositories on the worldwide web. These repositories Include:

- Kaggle Datasets (https://www.kaggle.com/datasets)
- Amazon Datasets (https://registry.opendata.aws/)
- UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/index.php)
- Google's Datasets Search Engine (https://toolbox.google.com/datasetsearch)
- Microsoft Datasets (https://msropendata.com/)
- Awesome Public Datasets Collection (https://github.com/awesomedata/awesome-public-datasets)
- Computer Vision Datasets (https://www.visualdata.io/)
- Government Datasets (https://data.europa.eu/euodp/data/dataset, https://www.data.gov/, https://catalogue.data.govt.nz/dataset, https://data.gov.in/, https://www.opendatani.gov.uk/)
- Lionbridge AI Datasets (https://lionbridge.ai/datasets/)

Data Analysis Methods:

Most of the datasets we've obtained were pre-formatted, cleaned and sorted. However, for improved efficiency, performance, and productivity we had to re-factor some of the pre-packaged datasets.

In case of the data collected under our responsibility, steps were taken to ensure that qualitative means of data collection was observed since tiniest of detail has a massive effect on the sensitivity of the program. The content analysis for the data we collected with the aid of xyz research center was done by practitioners (in this case, breast cancer and tuberculosis experts). Furthermore, a final dataset was prepared after carefully removing contamination from some gathered samples and the dimensions and size of each image was adjusted accordingly (keeping performance in mind).

Software Development Methods:

Feature-Driven Development method was best suited for our project considering features included in our project are client-valued pieces of work that, according to the FDD approach, should be distributed every a fortnight. FDD fixates on the goal of distributing working software frequently and is an especially client-centric approach, making it a good fit for more diminutive development teams. To engender tangible software often and efficiently, we followed FDD's five steps

1 Develop an overall model.
2 Build a feature list
3 Plan by each feature.
4 Design by feature
5 Build by feature

In every phase, reporting status was encouraged and helped to track progress, results, and possible errors.

1.6 Significance/ Potential Applications

With cancer, the key to successful treatment is detecting it early. As it stands, doctors have access to high quality imaging, and skilled radiologists can spot the telltale signs of abnormal growth.

Once identified, the next step is for doctors to ascertain whether the tumor is benign or malignant (in the case of breast cancer) or whether the cavitation in lungs is a sign of tuberculosis. Not everyone has access to such services. In most cases people fail to get an early diagnosis either due to financial troubles or a misdiagnosis or due to unavailability of experts.

This research aim was creating such an application out of this which aid a common person to get consultation from an automated system equivalent to hundreds of experts for such a low price or even for free. Even though it is evident that the use of Machine Learning methods can improve our understanding of cancer progression, an appropriate level of validation is needed in order for the methods used by the IRMI system to be

considered safe and reliable in everyday clinical practice. So, one potential and effective use of such an application would be early diagnosis before consulting a practitioner. This would not only save time and money of people and doctors but save lives as well; since doctors are potentially able to treat more patients with true positive illness.

2 LITERATURE REVIEW AND BACKGROUND

2.1 Background

Recent advances in medical imaging have transformed the analytical process in the medical field. Effective use of imaging data to enhance the diagnosis is highly in demand. The digitization of radiological imaging has opened up gates for data scientists to help the diagnosis process to not only make it more powerful, but more efficient and accurate. In practice, this can enable increase in productivity levels of doctors by decreasing the time required for image analysis while their cogency, in terms of precision and consistency, of diagnosing malignancy can be enhanced. Although an AI system can never replace an actual doctor but it can assist people by providing them with relevant information regarding the diagnosed disease and appropriate patient care.

Tons of research is already being carried out to integrate medical imaging into Artificial Intelligence for enhanced diagnosis. Many research papers have been written for effective ways to amalgamate AI and medical imaging for the purpose of disease detection but most of them never made into actual services that the world could use.

The purpose of this research is to not only to improve the accuracy of previously established researches but also on effective ways to integrate all these detection mechanisms into one single AI system for the benefit of humanity. An AI system which boasts the capabilities of handling multiple disease detection becomes a perfect application as an automated health care guide.

2.2 Literature Review

In the 1960s scientists made their first ever attempt to create a computer aided system for detection in image based data [1]. Since then the word has seen a variety of commercially deployed detection systems to be used clinically, including Riverain, CAD4 TB, and Delft imaging systems [2]. But due to the complexity of medical imaging, the automatic detection process has always been never been very accurate. Most of the existing CAD systems are aimed at the early detection of a certain illness detectable by medical imaging. Figuratively, very small number of studies are dedicated towards automatic detection of any illness diagnosable from medical imaging [3].

Nizar Banu PK's research paper is dedicated towards various experiments in the image processing and AI for the detection of TB in Chest X-Rays. The papers claims to have utilized novel method such as segmentation, filtering, extraction of features and demotion. Once an X-ray image is received input, pre-processing procedures like Gaussian and median filters are employed. These filters help to get rid of unnecessary noise and enable rich textures on the image. The output achieved from these are then passed through an additional gray level segmentation to properly obtain the lung area. Finally, the result obtained from the segmentation and filtration process is fused to obtain the maximum region of interest. From whcih, the statistical features like eccentricity, area, axises, average, skewness and entropy are calculated. To successfully detect the lung area sequential minimal optimization, linear regression and k-nearest neighbor are used in comparison. The paper concludes that k-nearest neighbor outperforms the other algorithms in detection of lungs in the chest area [4].

The early nineteen eighties saw an increase in use of artificial neural network on computer systems in the area of image processing. The diagnosis of breast carcinoma is very arduous. Therefore, mathematical statistics and artificial intelligence methodologies can be very useful in this matter [5]. Artificial Intelligence is known to be very useful in such detailed detection of pictorial data.

There is more than one way to screen breast carcinoma, such as: mammography, thermography and ultrasound. Computer aided detection based on image processing and artificial intelligence can help radiologist in early detection and accurate screening

of breast carcinoma. The authors of this publishing goes over multiple means of AI based detection using image processing to detect breast carcinoma. The authors had access to many well-known and large databases. In this study, more than 18,000 articles were mined between the years 2007 and 2017, included. Many of the articles published during this time used homogenous techniques and all of them used image processing. The feature list of image used within the article was also thoroughly listed. The conclusive results showed that support vector machine architecture gave the most promising results with highest precision over various images. Results: Mammograms: ^93%, Ultrasound: ^95%. [6].

Diagnosis of breast carcinoma and tuberculosis using computer based systems which greatly contributed to the development of early detection, are supportively being consumed by medical practitioners such as radiologist. It is considered ethical and useful in the field of medical with regard to its effects. Smart systems based on computers increase medical diagnosis precision by removing false positives.

2.2.1 Gap Analysis

Computer aided medical imaging systems generally follow these main steps for feature extraction: preprocessing the image, extracting regions of interests, extracting features from region of interest, and classifying a disease according to the features. But due to advancements in technology (hardware and software), scientists were able to process large amount of data within artificial intelligent systems. This opened up a variety of possibilities for computer aided detection systems since image based data requires a lot of processing power. Artificial intelligence based computer systems, especially deep learning, mainly supersede in the area of feature extraction compared to the traditional computer aided diagnosis systems. Artificial Intelligence based methods are additionally being widely utilized in image segmentation for example bone suppression of chest X-ray images. The machine learning methods are widely being used for classifying various diseases, but performance is dependent entirely on the features extracted. This is where deep learning offers a better insight since it provides better feature extraction performance.

3 SYSTEM MODELING & USE CASES

System Modeling:

Doctor Use Cases

Use cases Id: 01
Use case Name: Sign-Up

Use cases Id: 02
Use case Name: Login

Use cases Id: 03
Use case Name: Respond to Patient Request

Use cases Id: 04
Use case Name: View System Generated Result

Patient Use cases

Use cases Id: 05
Use case Name: Sign-Up

Use cases Id: 06
Use case Name: Login

Use cases Id: 07
Use case Name: Image Upload

Use cases Id: 08
Use case Name: View History

Use cases Id: 09

Use case Name: View Doctor List by Speciality

Use cases Id: 10

Use case Name: Select Doctor

Doctor's Use Cases:

Table 2.0 Signup

Use Case ID:	01
Use Case	Sign-up
Actors:	Doctor
Scope:	Doctors have to Sign-up.
Preconditions:	Enter user name. Enter password.
Post conditions:	Doctor cannot access his account if he enters a low case sensitive password or enters an incorrect username. If a Doctor enters Correct data then he/she can login for the App.
Normal Flow:	A Doctor can register him/herself.
Main Success Scenario:	He can be able to Login
Assumptions:	Doctor can login.
Exceptions:	None.

Description: This table represents signup use case, in our system.

Table 2.1 Login

Use Case ID:	02
Use Case	Login
Actors:	Doctor
Scope:	Doctors can have to Login to access app.
Preconditions:	Doctor Must have an account.
Post conditions:	If he enters correct data then he can redirect to the App front Page.
Normal Flow:	Doctor opens IRMI app. Select account type. Enter the correct username. Enter the correct password. If the username or password is wrong then the error message shows. Doctors can be directed to the App.
Main Success Scenario:	Doctors can access App.
Assumptions:	None.
Exceptions:	Doctor cannot access his/her account if he/she enters an incorrect username or Password.

Description: This table represents login for doctor, use case in our system.

Table 2.2 Respond to Patient Query

Use Case ID:	03
Use Case	Respond to Patient Query
Actors:	Doctor
Scope:	A Doctor can choose a patient to respond at a single time.
Preconditions:	Patients can try to contact the doctor.
Post conditions:	Doctor can be in contact with the Patient.
Normal Flow:	A Doctor can review the results and can write his/her opinion.
Main Success Scenario:	A Doctor can respond to the patient's query.
Assumptions:	A Doctor can see the result.
Exceptions:	None.

Description: This table represents response to patient query from doctor, use case in our system.

Table 2.3 View System Generated Result

Use Case ID:	04
Use Case	View System Generated Result
Actors:	Doctor
Scope:	A Doctor can see Patients result and can download them.
Preconditions:	Doctor must have to select the patient to see the result.
Post conditions:	A Doctor can give prescriptions to patients.
Normal Flow:	A Doctor can see and analyze the result and can set private meetings for patients.
Main Success Scenario:	A Doctor can analyze the X-ray.
Assumptions:	A Doctor must be relative to the field.
Exceptions:	None

Description: This table represents viewing of system generated results on doctor's side, use case in our system.

Patient Use Cases:

Table 2.4 Sign-Up

Use Case ID	05
Use Case	Sign-Up
Actors:	Patient
Scope:	Patients have to Sign-up.
Preconditions:	Enter user name. Enter password.
Post conditions:	Patient cannot access his account if he enters a low case sensitive password or enters an incorrect username. If the Doctor enters Correct data then he/she can login for the App.
Normal Flow:	Patients can register him/herself. They are be able to Login
Main Success Scenario:	None.
Assumptions:	A doctor must be of relative field.
Exceptions:	If the patient name is in incorrect form he has to enter the correct username.

Description: This table represents signup for patient, use case in our system.

Table 2.5 Login

Use Case ID	06
Use Case	Login
Actors:	Patient
Scope:	Patient can have to Login to access the app.
Preconditions:	Patient Must have an account.
Post conditions:	Patient cannot access his account if he enters an incorrect username or Password. If he enters correct data then he is redirected to the App front Page.
Normal Flow:	Patient opens IRMI app. Enter the correct username. Enter the correct password. If the username or password is wrong then an error message shows. Else patients are directed to the App.
Main Success Scenario:	Patients can access the App.
Assumptions:	None.
Exceptions:	You must have sign-up once.

Description: This table represents login for patient, use case in our system

Table 2.6 Image Upload

Use Case ID:	07
Use Case Name:	Image Upload
Actors:	Patient
Scope:	Patient can have to upload images and our App can detect the disease.
Preconditions:	Patient must be logged into the app.
Post conditions:	Patient can save the X-ray.
Normal Flow:	Patient can upload his image. Then he is able to get the result and show it to a specific doctor.
Main Success Scenario:	When a patient uploads an image he gets output of the result.
Assumptions:	
Exceptions:	If the image is not correct like a blurry picture or not related to our app demand then the system gives error.

Description: This table represents image upload form patient, use case in our system.

Table 2.7 View History

Use Case ID:	08
Use Case Name:	View History
Actors:	Patient
Scope:	Patient is able to see the history result if needed any time.
Preconditions:	Patient already saved the result in view of the results history.
Post conditions:	Patient results are saved in view history.
Normal Flow:	Patient can choose any result he/she wants to open and can see it.
Main Success Scenario:	Patients all results are saved in the folder.
Assumptions:	None
Exceptions:	If a patient has no history records he cannot be able to see.

Description: This table represents viewing history as patient, use case in our system

Table 2.8 View Doctors List by Specialist

Use Case ID:	09
Use Case	View Doctors List by Specialist
Actors:	Patient
Scope:	A list of doctors is shown and patients can select a doctor according to his/her disease specialist.
Preconditions:	Patient must have a result.
Post conditions:	Patients can contact the doctor.
Normal Flow:	Patients can see available doctors.
Main Success Scenario:	Patient uploads the result so he can select a specialist doctor.
Assumptions:	None
Extensions:	None.

Description: This table represents viewing doctor list specialist by patient.

Table 2.9 Select Doctor

Use Case ID:	10
Use Case	Select Doctor
Actors:	Patient
Scope:	Patient can get the recommendation on the basis of his result.
Preconditions:	Patient selects the doctor after checking the ratings.
Post conditions:	A Doctor can be able to see results or can give his/her opinions to patients.
Normal Flow:	Patients can select a doctor on the basis of rating and he/she can choose doctor to the relative disease.
Main Success Scenario:	Patient becomes in contact with the doctor.
Assumptions:	None.
Exceptions:	None.

4 PROPOSED METHODOLOGY

4.1 Suggested Approach

Early detection and competent treatment of breast cancer or tuberculosis may increment treatment chances and reduce mortality rates. Recent clinical studies stipulate that computer aided diagnosis systems have a huge importance in raising the survival rates of affected patients i.e. through early detection.

From previous research we've learned that the best approach for successful detection is to perform image processing on the raw data for highlighting certain aspects of the images, reduction of background by using combinations of linear and non-linear filters, compiling the images after lowering their resolution(higher efficiency) into datasets, and finally creating a model for that specific detection type.

From previous research, about the field of breast cancer detection in mammograms, the suggested approach to image pre-processing utilized by Hao-Chun Lu and El-Wui Loh [1], CLAHE (Contrast Limited Adaptive Histogram Equalization) is promising. This approach provided us with the most clear and highlighted results for the pre-processing stage. It should also be noted that mammograms are highly detailed. Hence, working with lower resolution images is not an option here since it results in very high inaccuracy as demonstrated in experimentation. This approach increases the time it takes for the image classification process (discussed in the Experimentation section). We have also demonstrated the benefits and drawbacks of performing the additional image processing by converting the images into heatmaps.

In the case of tuberculosis, we trained a Fine-Tuned Neural Network VGG16 (Karen Simonyan & Andrew Zisserman[7]) model with TensorFlow's Keras API.

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual computer vision competition. Annually, the competitors compete on two assignments. First assignment being able to detect objects within pictorial data anticipated from 200 classes, which is known as localization of object. Second, classifying the obtained images (labelled) within 1000 available categories. A new architecture: VGG 16, was

put forth by Andrew Zisserman and Karen Simonyan of the Visual Geometry Group Lab at Oxford University in the year 2014 published in the research paper "VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION". In this competition, this model won first and second place in the above listed categories.

4.2 Workflow of the system

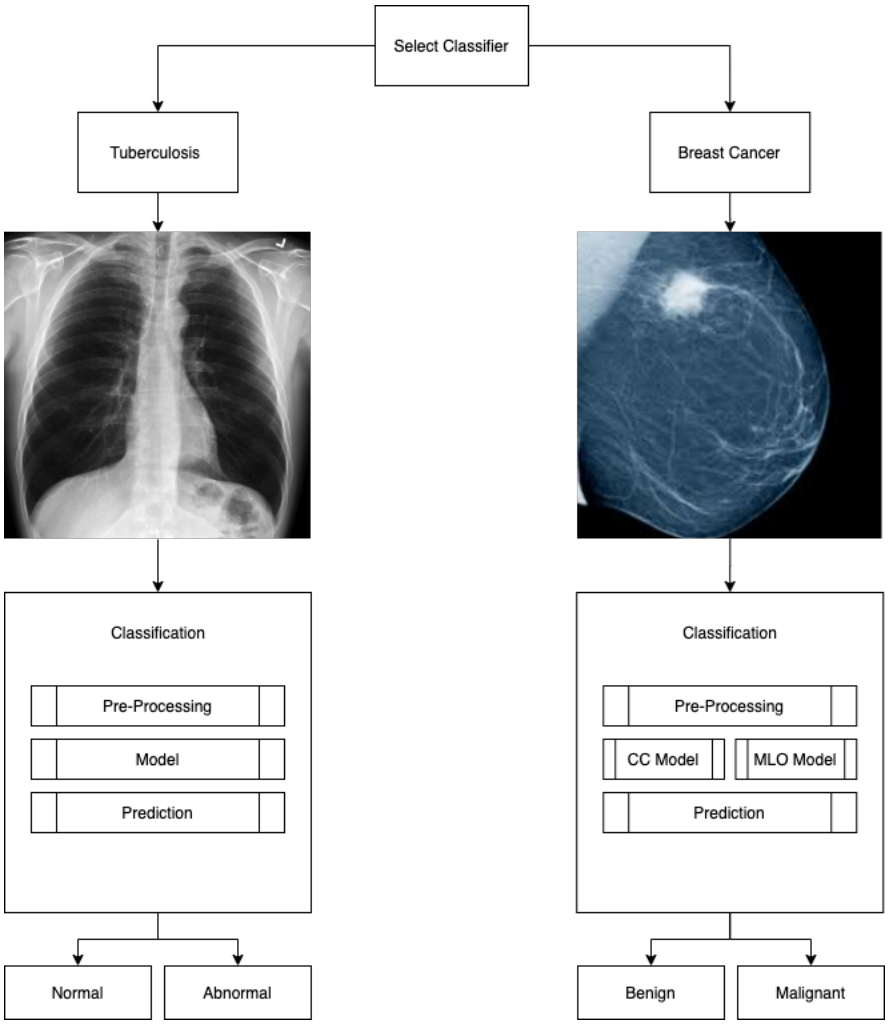


Figure 1. IRMI System

This figure represents workflow of our system as a whole where initially image is under per-processing then it enter in the model stage and then predict the disease whether it is normal or abnormal.

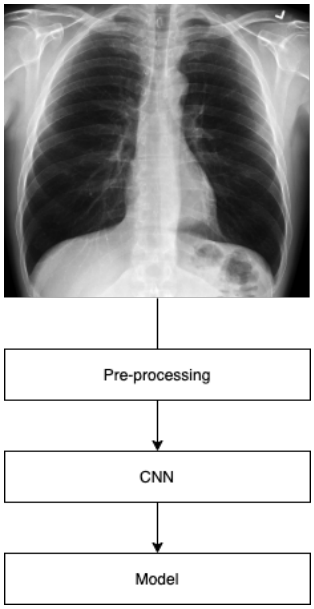


Figure 2. Tuberculosis Model

This figure represents classification process for tuberculosis detection in our system. In it, image processing and then comes under the CNN model and then it is classifier.

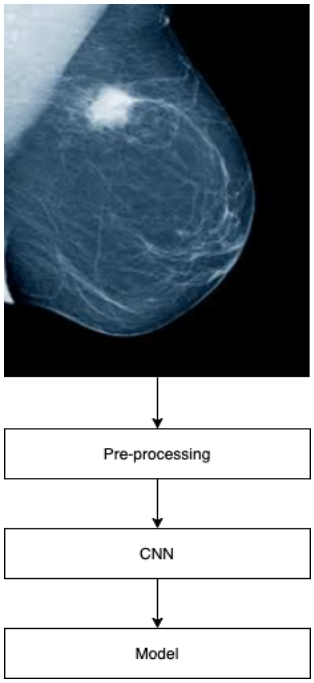


Figure 3. Breast Cancer Model

This figure represents classification process for breast cancer detection in our system. In it, image processing and then comes under the CNN model and then it is classifier.

Environment setup

The datasets and models were created/compiled on a Mac based computer. The computer's specifications were: Intel core i7 7700k (5.5 Ghz), 16GB of DDR4 RAM, 384 GB Solid state drive, and an NVIDIA 1080 based graphic processing unit with 1.5 GB of graphic memory.

Core settings

In each case, we chose 25% of the total of the dataset was used for testing the data, 10% of the dataset was used for the validation of the data and 65% of the dataset was used for the training of the data. Keras and TensorFlow (TF) frameworks were used to create custom architectures for testing the efficiency of the models.

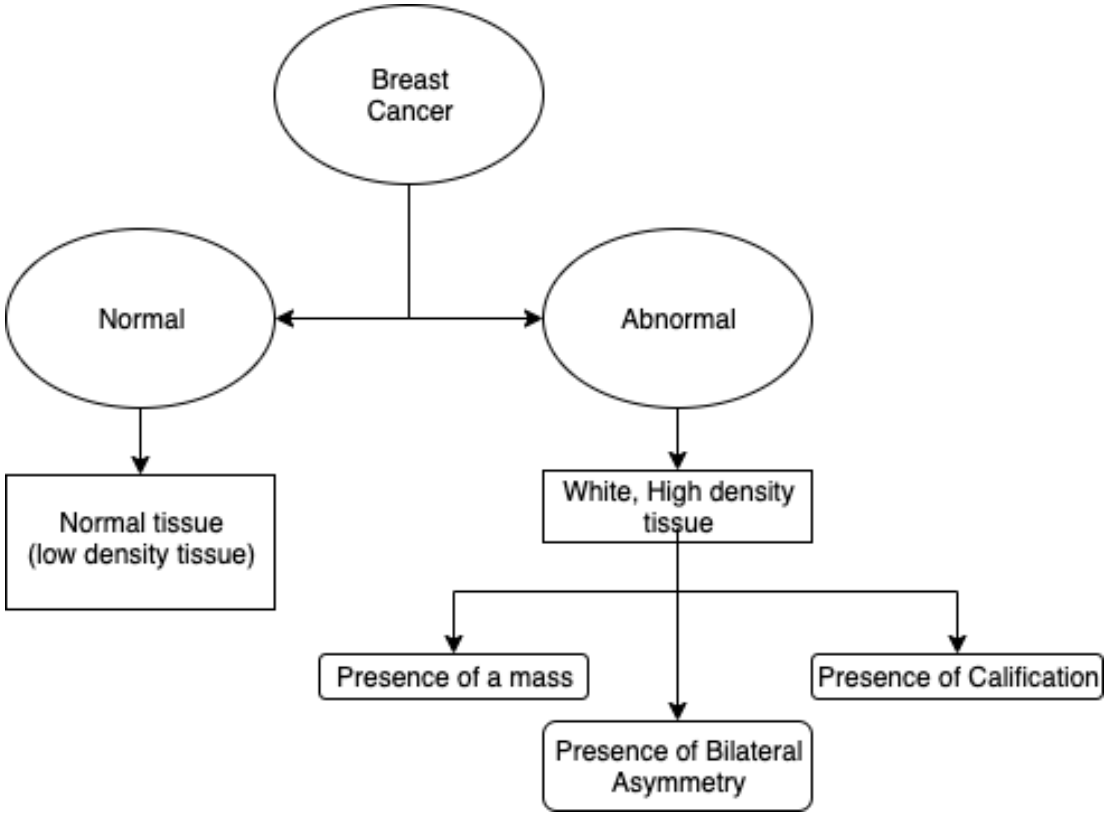


Figure 4. Breast cancer classification

This figure represents classification process for breast cancer

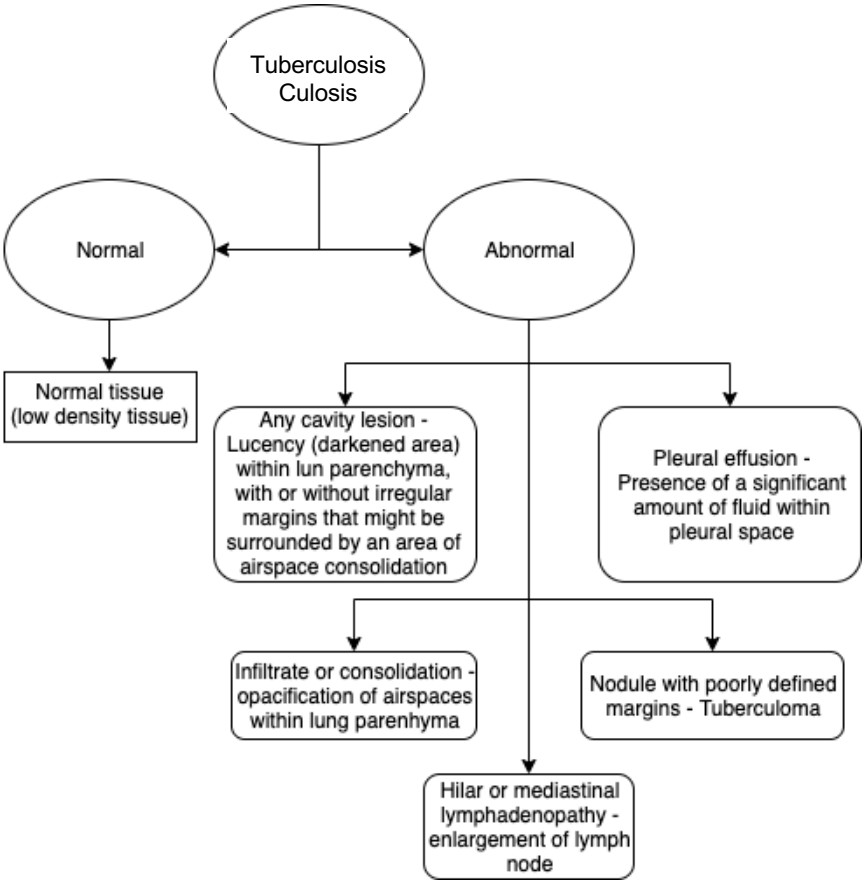


Figure 5. Tuberculosis classification

This figure represents classification process for tuberculosis.

4.3 Algorithms/Architecture

Preprocessing

In the case of Tuberculosis, the grayscale chest radiographic 224x224 matrix while their format was kept as PNG (Portable Network Graphics). We used median filter because it gave the best results when it came to image sharpening, removing noise and highlighting features of the images. Canny edge detection algorithm was applied to lung images to detect lung area.

In the case of breast cancer, because the mammograms are highly detailed and quality of the image contributes towards better classification, we opted for higher resolution images. We use an input resolution of 2677 × 1942 pixels for CC views, and 2974 ×

1748 pixels for MLO views. The high-resolution images are then cropped to remove the unnecessary data. In order to enhance the mammogram, we used CLAHE (Contrast Limited Adaptive Histogram Equivalization) for increasing the contrast. Adaptive histogram equalization (AHE) improves on this by transforming each pixel with a transformation function derived from a neighborhood region. From the enhanced images, augmented 256 × 256 patches are used as inputs to the patch classifier. Ultimately, we send the images to the model based on the orientation and view of the image (CC and MLO respectively).

Filtering methods have been used in both mammogram and chest X-rays to reduce the noise.

Model

After pre-processing is applied the gathered dataset is split into 80% train and 20% test data. The type of model used is sequential which a linear stack of layers is.

Classification

All images were augmented by using random cropping of 256 x 256 pixels patches, mean subtraction, and mirror images which were pre built options within the TF framework. Further augmentation was performed in training some of the Deep Convolutional Neural Network (DCNN), including rotations of 90`, 180` and 270`, while most of training was performed on CLAHE processed images. We trained this network with Adam optimizer with a learning rate 0.0001 and Rectified Linear Unit (ReLU) activation function and dropout rate 0.5. The architectures for different models are illustrated in the table below:

Table 1. Architecture

	KERNEL SIZE	STRIDES	FILTERS
CONVOLUTION	3*3	2*2	32
MAX_POOLING	3*3	3*3	32
CONVOLUTION	3*3	2*2	64
CONVOLUTION(REPEAT*2)	3*3	1*1	64
MAX_POOLING	2*2	2*2	64
CONVOLUTION(REPEAT*3)	3*3	1*!	128
MAX_POOLING	2*2	2*2	128
CONVOLUTION(REPEAT*3)	3*3	1*2	256
MAX_POOLING	2*2	2*2	128
VCONVOLUTION(REPEAT*3)	3*3	1*2	256
FC NEURAL NETWORK		1024	
FC NEURAL NETWORK		2	

Description: This table represents model's architecture of our system.

5 DESIGN AND IMPLEMENTATION

5.1 System Design

The various system design issues that have to be addressed are:

Performance:

Our aim was to achieve an accuracy of 90% or higher in breast cancer on mammogram and Tuberculosis in Chest X-rays.

Robustness:

Noise is a fundamental characteristic that is present, to some extent, in all images. It reduces the visibility of some structures and objects, especially those that have relatively low contrast. In medical imaging the objective is not to eliminate the noise, but to reduce it to a clinically acceptable level. Our system has the capability to detect and manage noisy images, given as input.

Interactively:

Our system provides simplicity, speed, search options, bright color schemes, push notifications, user feedback and updates to better the user experience and interaction with web and mobile (IOS and Android) applications.

Reusability and portability:

Our system focuses on reusability in regards to the development goal of achieving low cost, high speed and quality.
A computer program is portable if it can be used in an operating system other than the one in which it was created without requiring major rework. Porting is the task of doing any work necessary to make the computer program run in the new environment.
The same code with modifications can be used for detection of different diseases aside from breast cancer and tuberculosis while providing the usability of the same software in different environments.

Security:

Internet security is the first thing where we cannot compromise. Multiple hackers from the globe spread their tricks and destroy the data. They may attempt the following activity:

- They may put some irrelevant data into apps and onto devices. Further they can access data and steal screen lock passcodes.
- Sensitive information traveling over the network.
- Steal customer data for identity theft or fraud.
- Get hold of private business assets

All of those potential issues mean that your app security must be watertight, rather than something that you tack on as an afterthought.

The system makes sure that user's data is secured and it is not shared with some third party without user's approval.

Architecture Design:

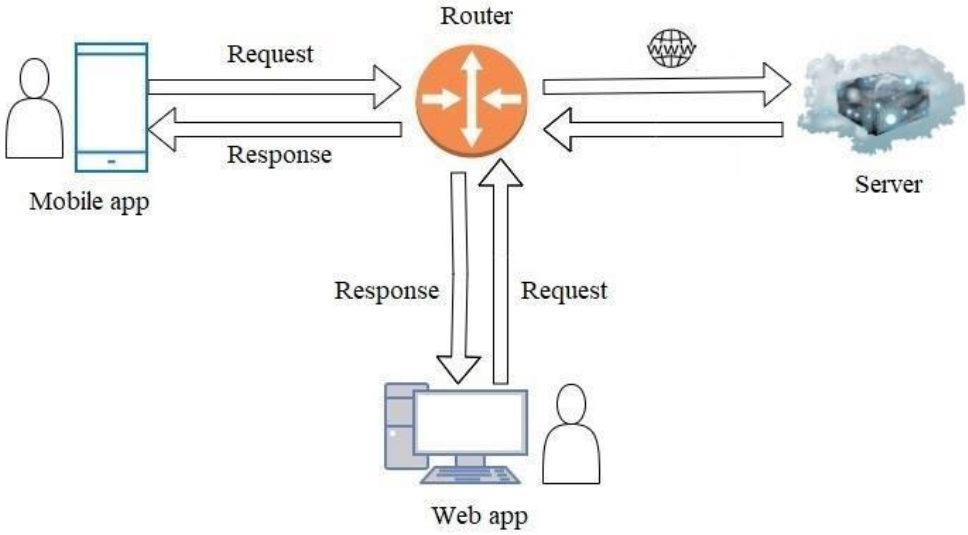


Figure 6. Architecture Design

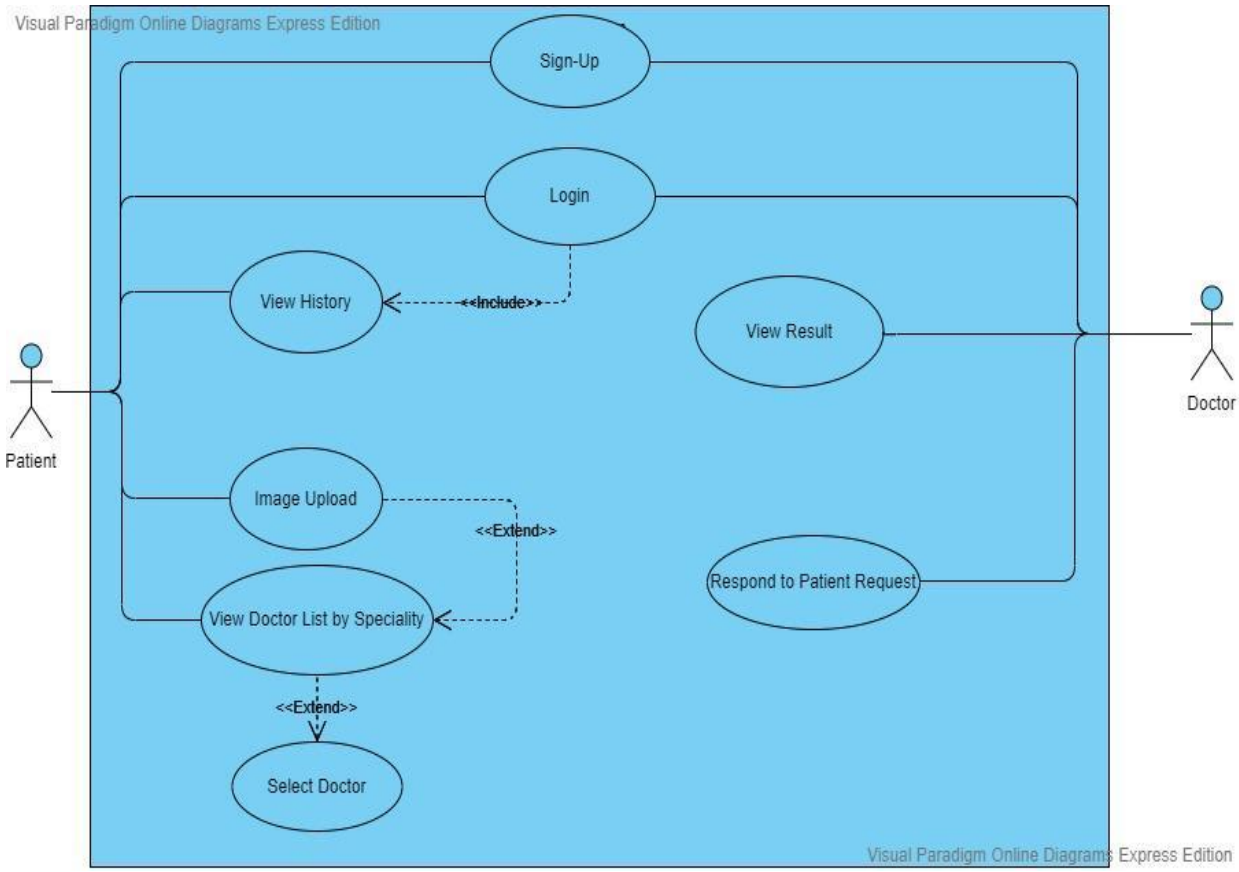


Figure 7. Actors and Use Cases

This figure represents architectural design of our system

Interface Design:

Our system provides simplicity, speed, search options, bright color schemes, push notifications, user feedback and updates to better the user experience and interaction with web and mobile (IOS and Android) applications.

Database Design – ERD diagram

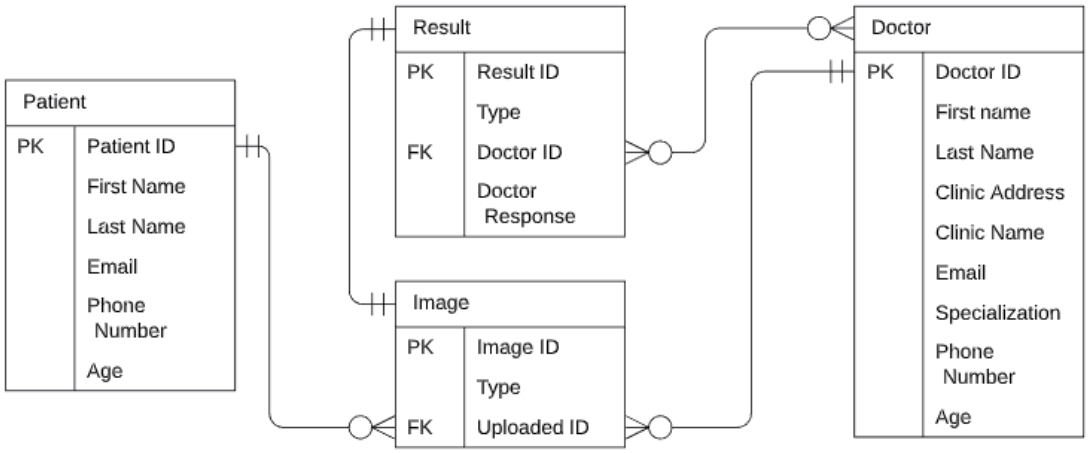


Figure 8. Entity Relationship Diagram

This figure represents Entities Relationship in our system

5.2 System Implementation

Develops tools and environment used:

Tools and languages:

1. JavaScript
    - Node
    - React
    - React native
2. Python
    - Jupiter
    - Flask

Environments used for development:

1. Mac OS
2. Windows

Sample Code:

```
import os
import base64
import io
from PIL import Image
import urllib.request
import mammograms
import tuberculosis
import time
import json
from flask import Flask, request, redirect, jsonify, Response, render_template
from flask_cors import CORS
from werkzeug.utils import secure_filename

UPLOAD_FOLDER = 'sample_single_input/'
ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg'])

app = Flask(__name__)
# cors = CORS(app, resources={r"/api/*": {"origins": "*"}})
#app.secret_key = "secret key"
CORS(app)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
selected_view = "L-CC"

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/file-upload', methods=['POST'])
def upload_file():
    # check if the post request has the file part
    if 'file' not in request.files:
        resp = jsonify({'error' : 'No file part in the request'})
        resp.status_code = 400
        return resp
    file = request.files['file']
    if file.filename == '':
        resp = jsonify({'error' : 'No file selected for uploading'})
        resp.status_code = 400
        return resp
    if file and allowed_file(file.filename):
        # filename = secure_filename(file.filename)
        selected_view = request.form.get("select-view")
        print(selected_view)
        filename = 'mammogram.png'
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        resp = jsonify({'success' : 'File successfully uploaded'})
        resp.status_code = 201
        return resp
    else:
        resp = jsonify({'error' : 'Allowed file types are png, jpg, jpeg'})
        resp.status_code = 400
        return resp

@app.route('/api/bc')                    #API for Breast Cancer classification
def progress():
    def generate():
        x = 0
        x = x + 10
        yield "data:" + str(x) + "\n\n"
        mammograms.crop_mammogram(selected_view)
        time.sleep(2)
        x = x + 20
        yield "data:" + str(x) + "\n\n"
```

```
        mammograms.calc_optimal_centers()
        time.sleep(2)
        x = x + 30
        yield "data:" + str(x) + "\n\n"
        mammograms.generate_heatmap()
        x = x + 30
        yield "data:" + str(x) + "\n\n"
        resp = mammograms.classify_mammogram(selected_view)
        time.sleep(2)
        x = 100
        yield "data:" + str(x) + "\n\n"
        yield "data:" + resp + "\n\n"

    return Response(generate(), mimetype= 'text/event-stream')

@app.route('/api/tb', methods=['POST'])     #API for Tuberculisis classification
def predict():
    message = request.get_json(force=True)
    encoded = message['image']
    decoded = base64.b64decode(encoded)
    image = Image.open(io.BytesIO(decoded))
    processed_image = tuberculosis.preprocess_image(image, target_size=(224, 224))

    prediction = tuberculosis.model.predict(processed_image).tolist()

    response = {
        'prediction': {
        'abnormal': prediction[0][0],
            'normal': prediction[0][1]
        }
    }
    return jsonify(response)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/tb')
def tb():
```

Mobile application:

Figure No: 9 Mobile Home Screen

This is home screen for registration or login.

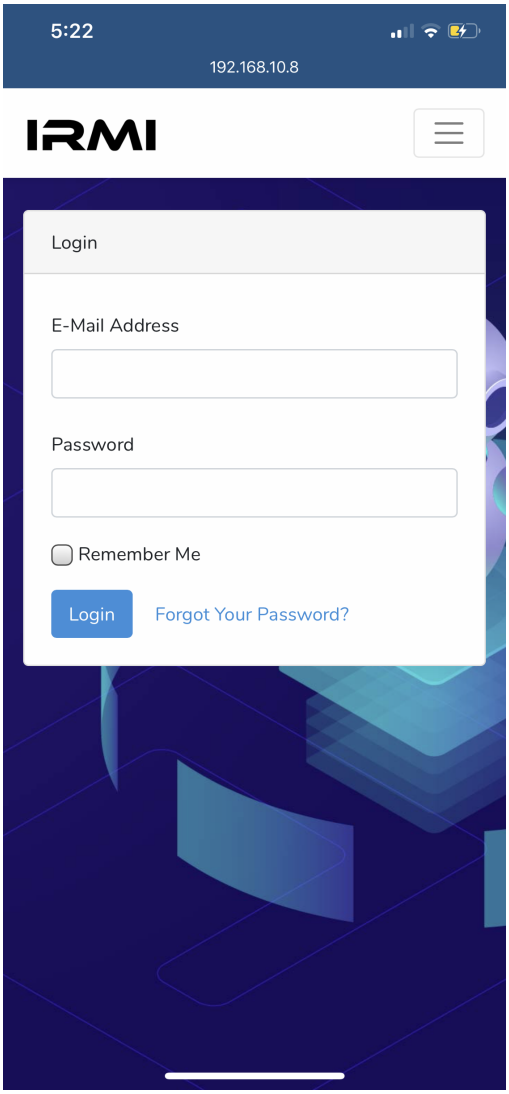


Figure No: 10 Mobile Login

This is mobile login Page.

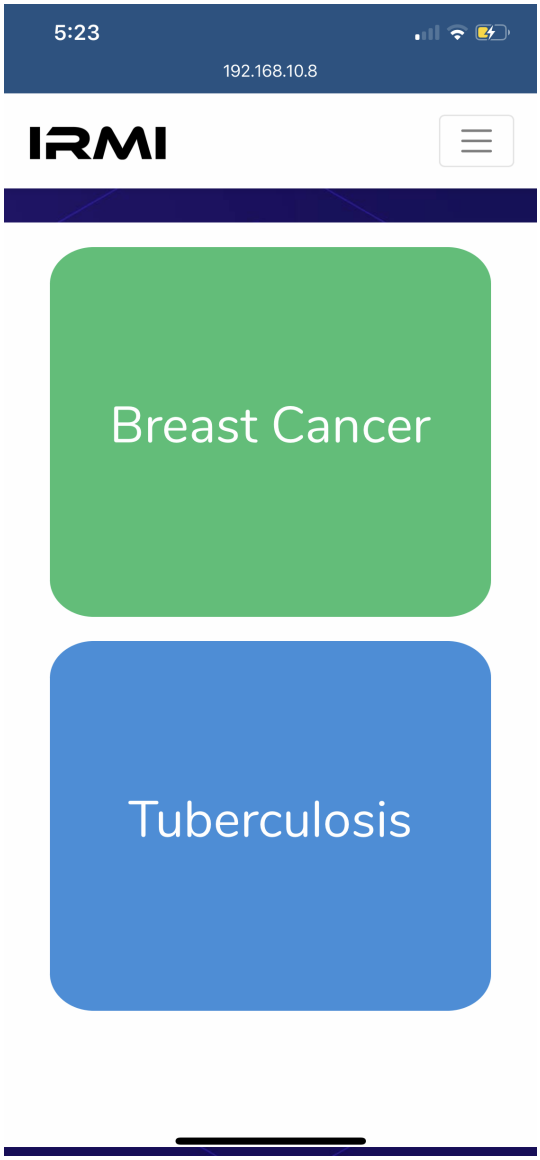


Figure No: 11 Mobile Welcome Screen

It is a mobile welcome screen for user.

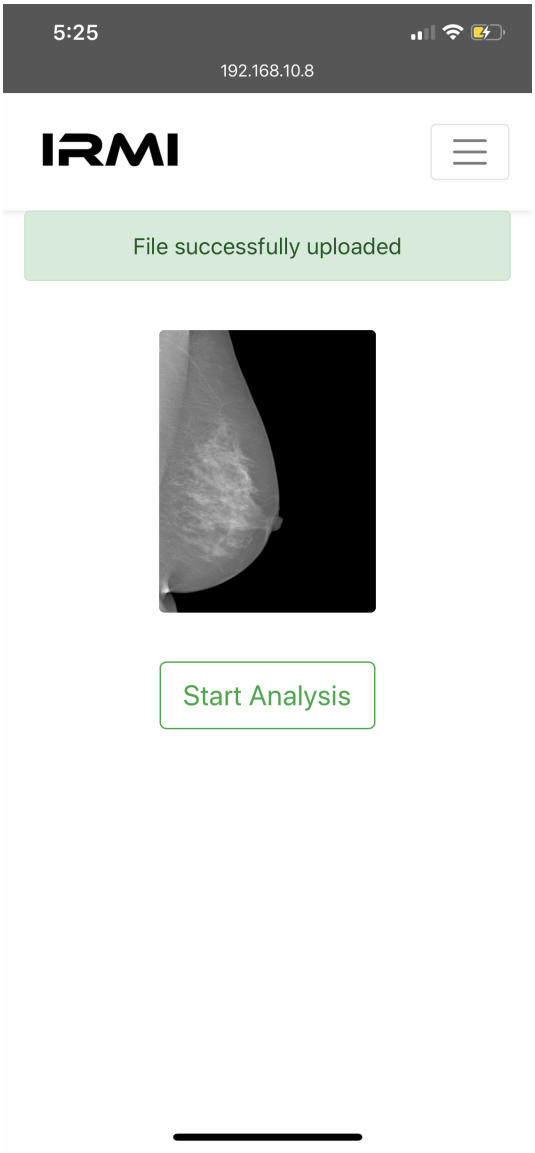


Figure No: 12 Mobile Breast Cancer Classifier

This is image successfully uploaded.

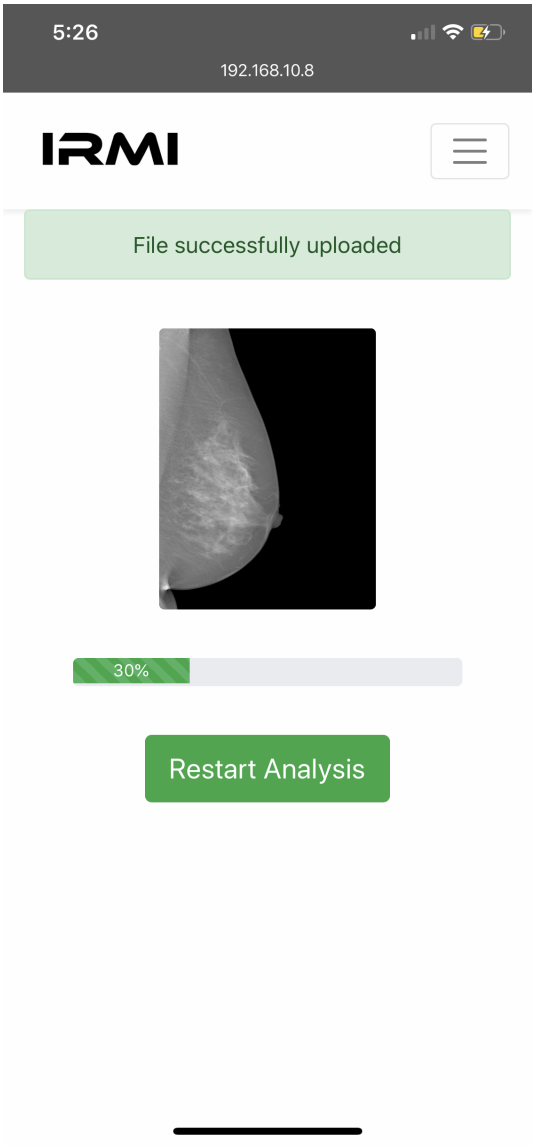


Figure No: 13 Mobile Breast cancer Processing

Image is processing in this picture.

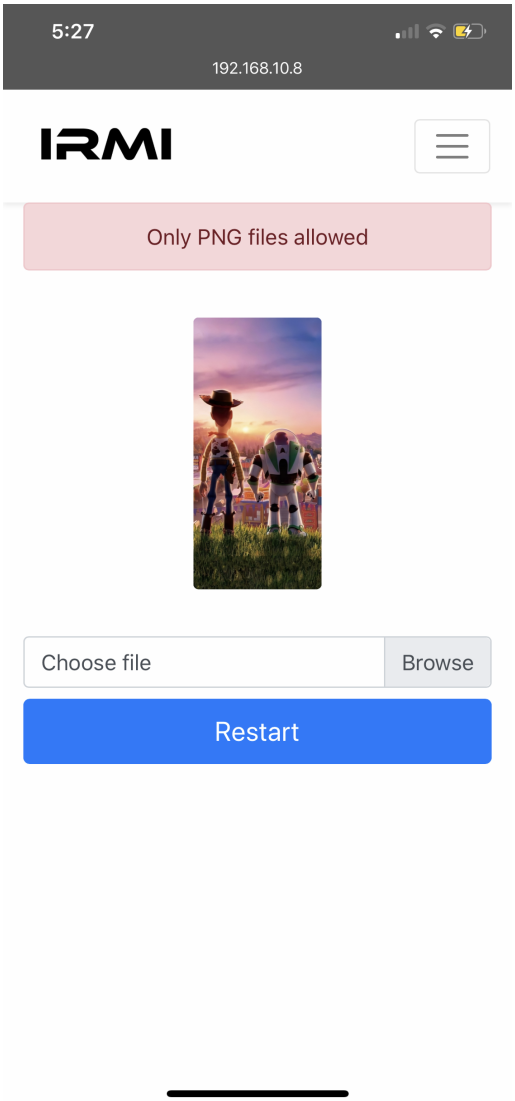


Figure No: 14 Mobile Format Picture

Only specified pictures can be uploaded.

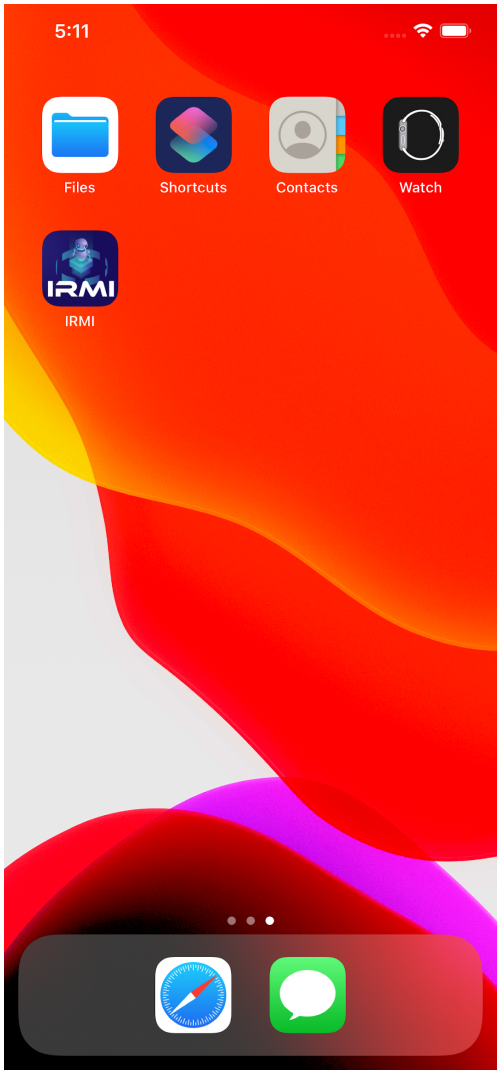


Figure No: 15 Mobile App Icon

This is the mobile home screen showing App Icon.

Web application:

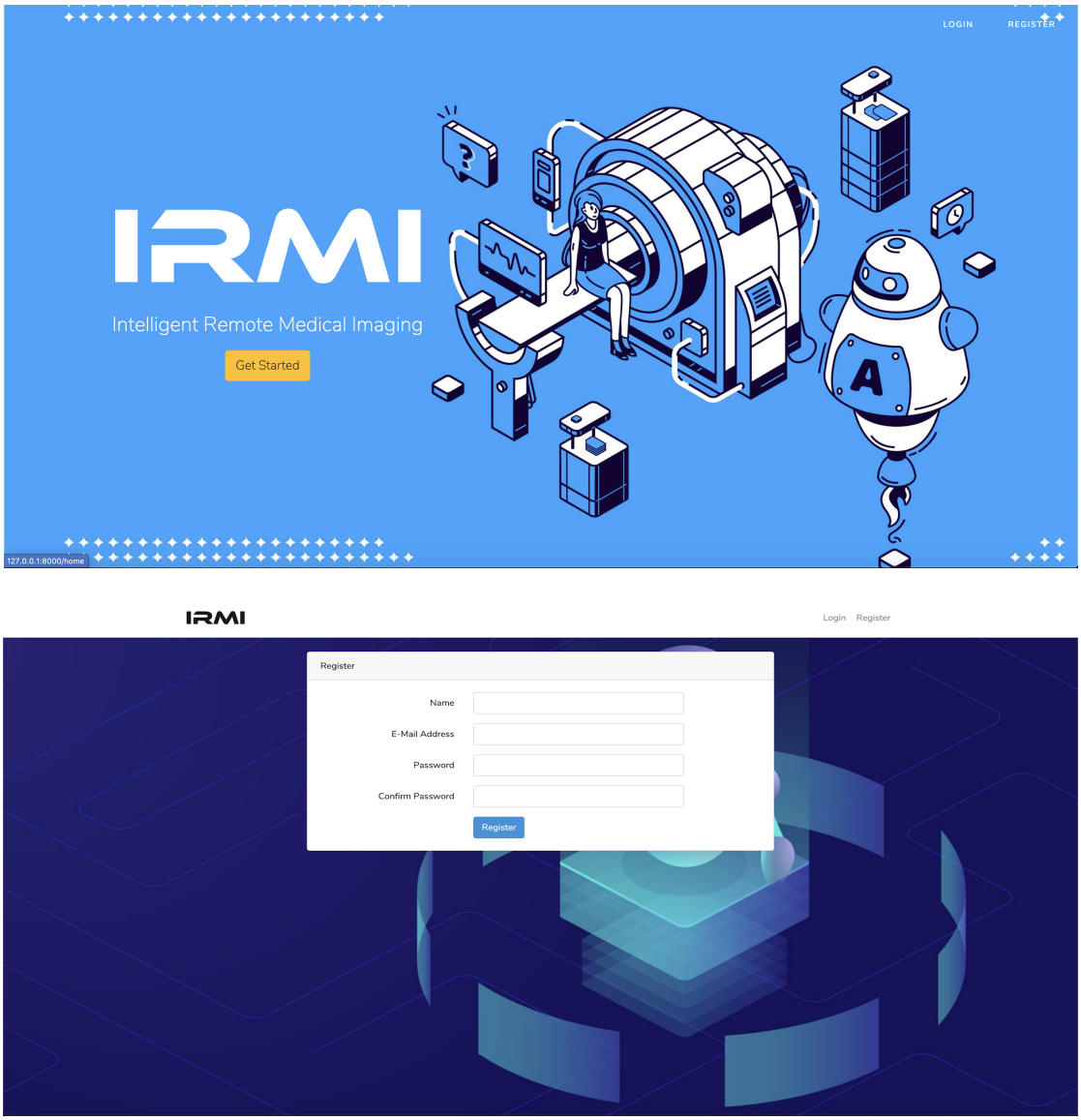


Figure No: 16 Web App Signup & Home Screen

This is the web app login/signup page.

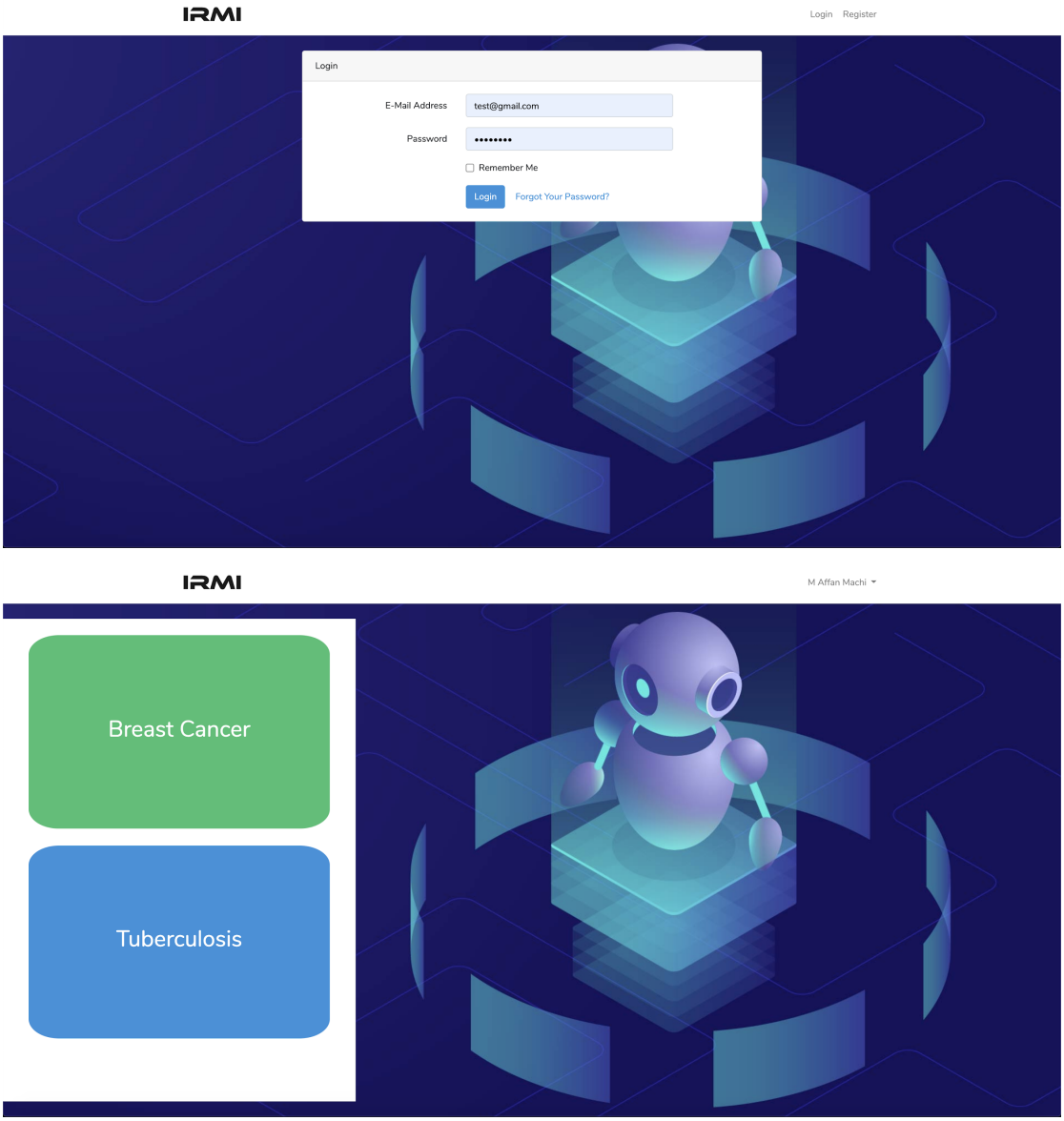


Figure No: 17 Web Login & Welcome

This is mobile login & welcome page.

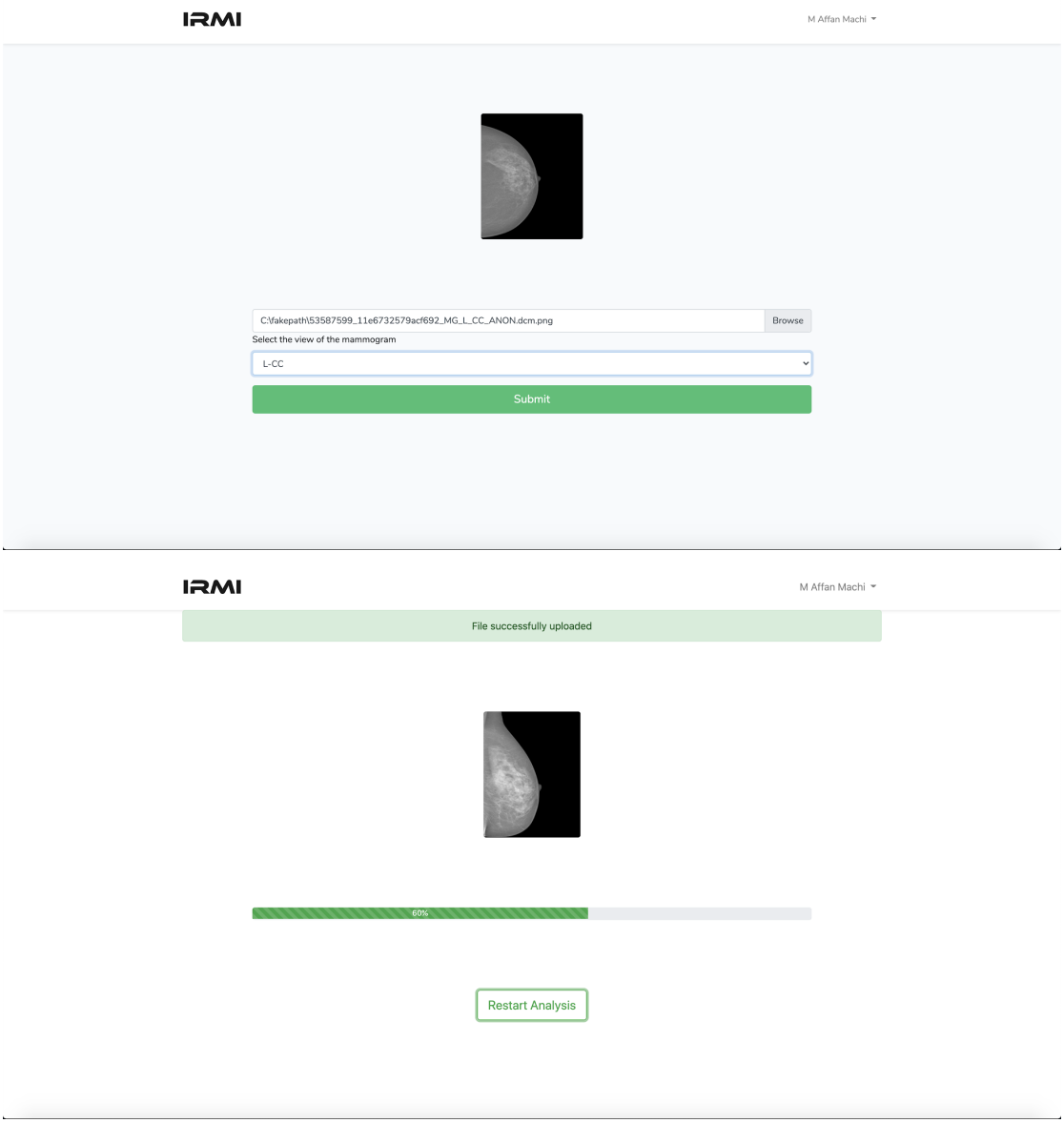


Figure No: 18 WebApp Breast Cancer Classification Process

This is Web app file upload and image processing display.

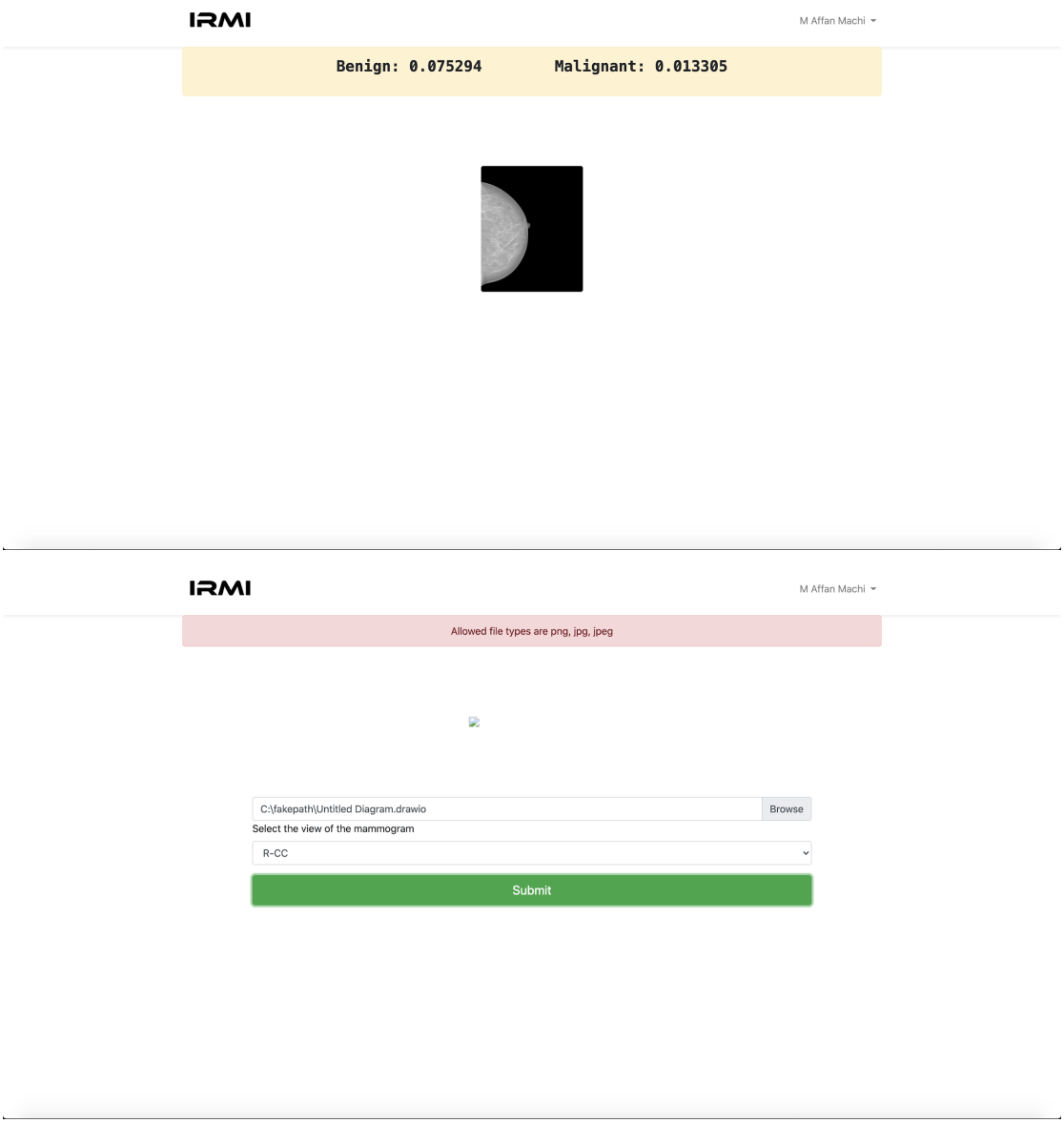


Figure No: 19 Breast cancer Classification Result & Error

This is the result and error showing page.

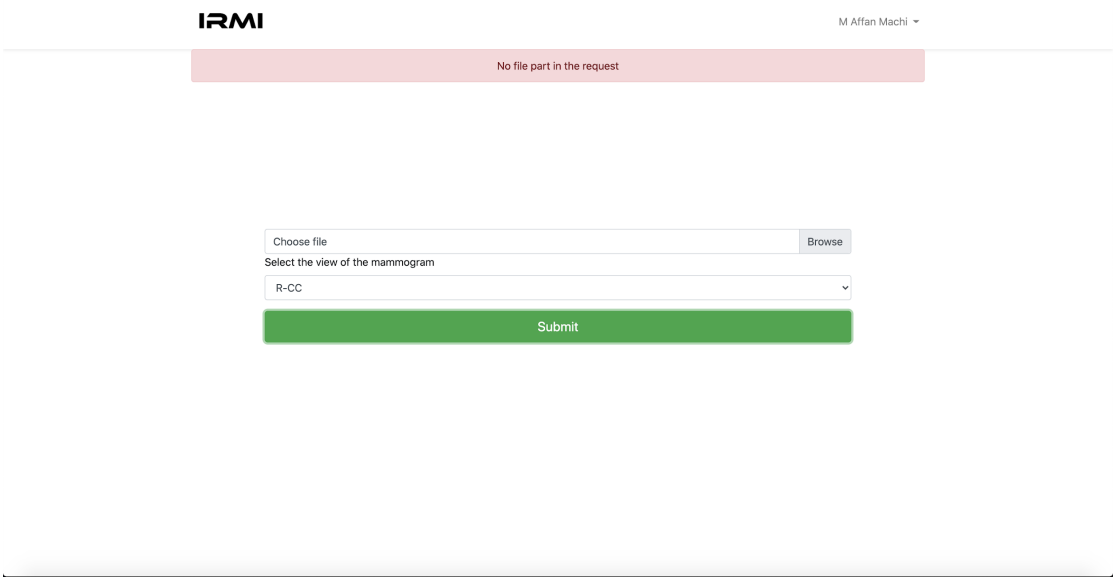


Figure No: 20 Web Breast Cancer Classification Errors

It is showing no picture is uploaded.

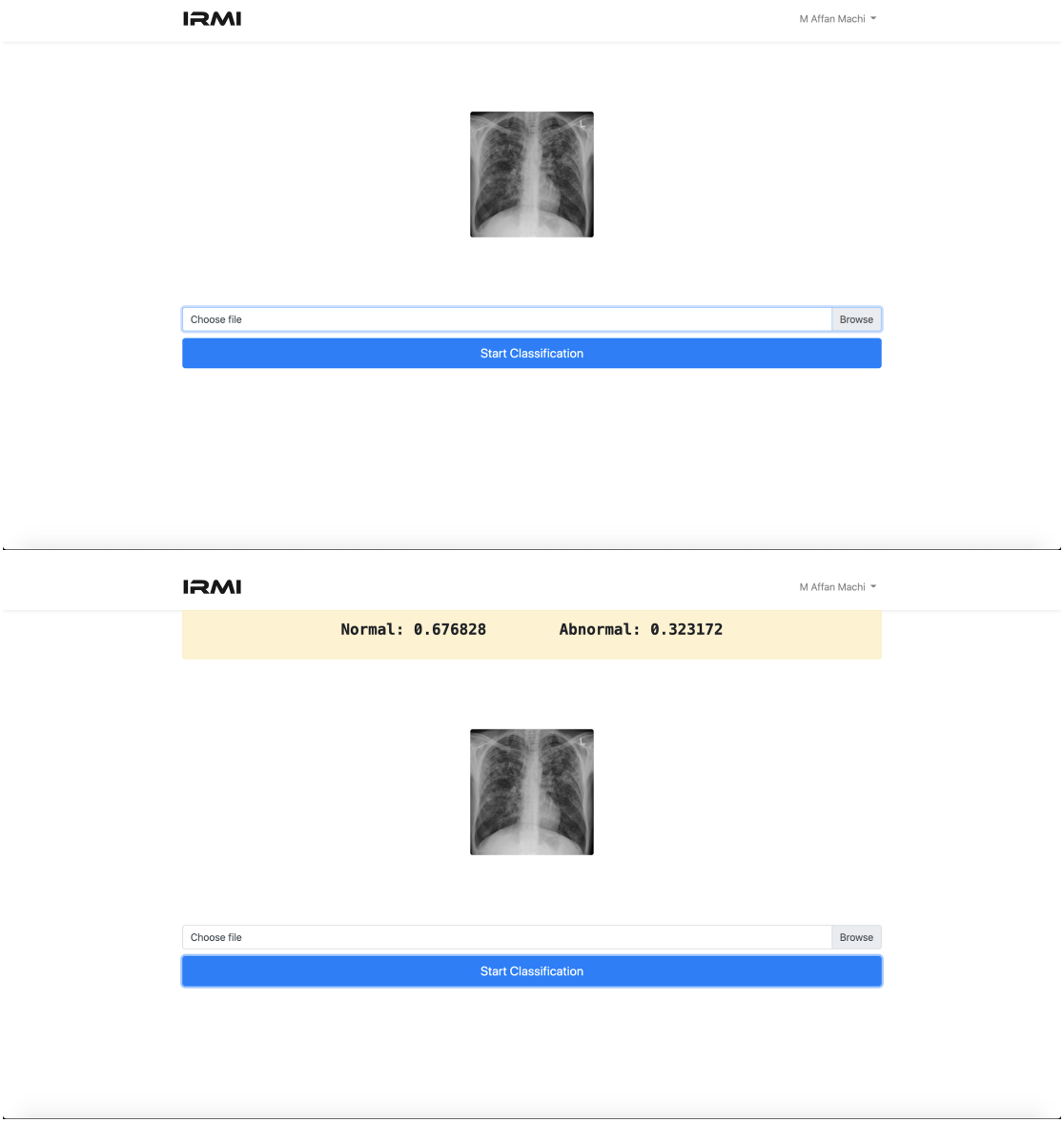


Figure No: 21 Web Tuberculosis Classification Process

It is file upload page and result showing page.

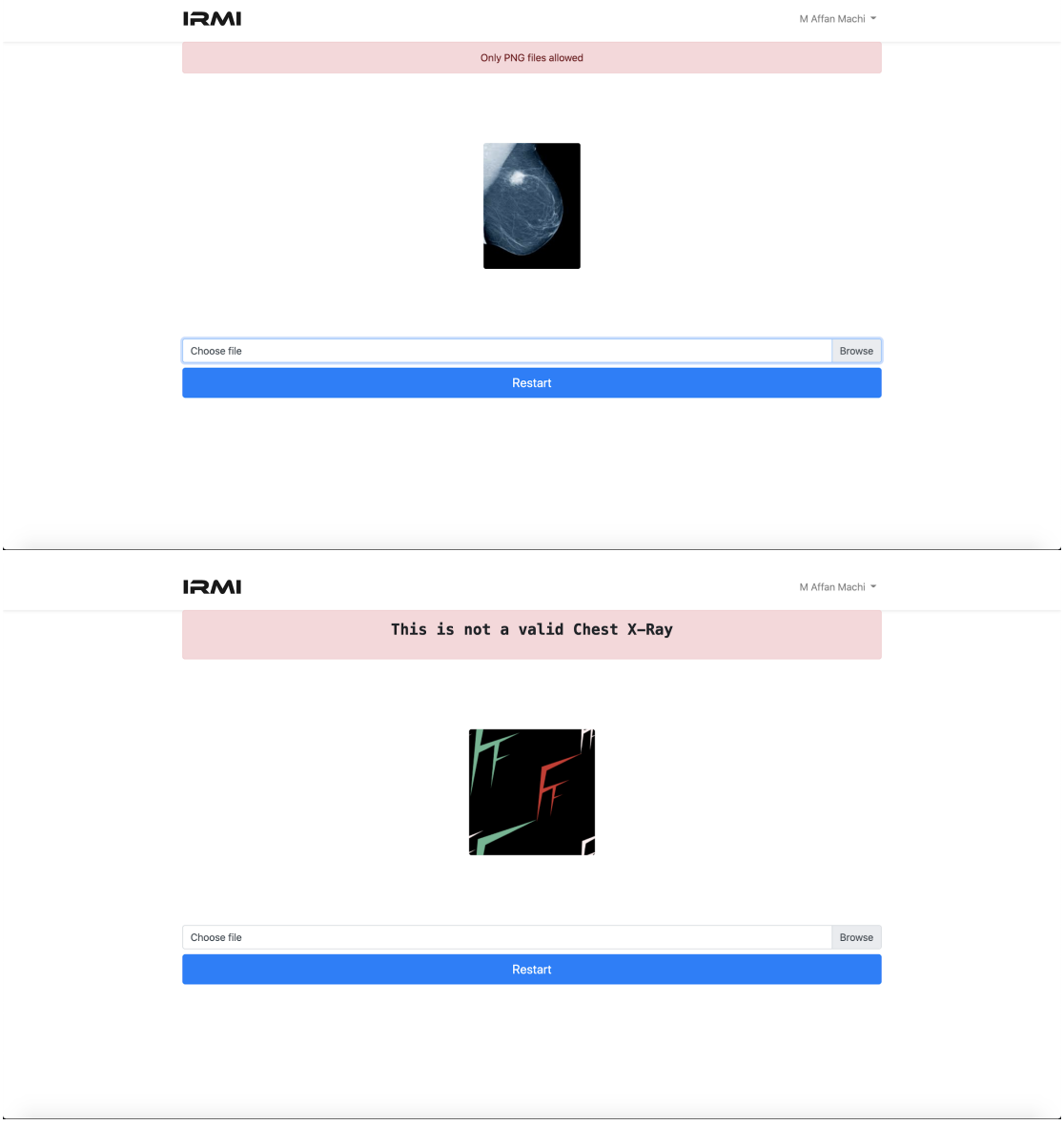


Figure No: 22 Web Tuberculosis Errors

Only specific picture is allowed and wrong image is uploaded.

Difficulties faced and how they were addressed

The changes required by TensorFlow in newer versions now require you to pass in a new parameter to the function 'fit()'.

You must pass steps_per_epoch parameter to the function 'fit()' if dataset is an infinitely repeating dataset

Depending on the version 2.2 or later of TensorFlow, you must specify this parameter to the function 'fit()' of your model. Else, you can get stuck on the first epoch infinitely. Because we are using Tensorflow 2.2 we specified the steps_per_epoch parameter because it is required in the later versions of Tensorflow.

5.3 Assumptions/Constraints

One assumption of this study is that the model, in the future the amount of data available data will increase. A large-scale, breast cancer mammogram and tuberculosis chest X-ray database will help to create more generalized models which will be more accurate.

6 EVALUATION

6.1 Experimentation

All experiments utilized the training data for streamlining boundaries of our model and the validation data for tuning hyper parameters of the model and the preparation methodology. Except if in any case indicated, results were registered over the screening populace. We used three different types of activation functions in CNN which are known for their best performance:

- ReLU
- SoftMax
- SoftPlus

We tried multiple iterations of the following experimental model. But optimum points and prediction stats were achieved by the following setup.

6.1.1 Experimental Setup

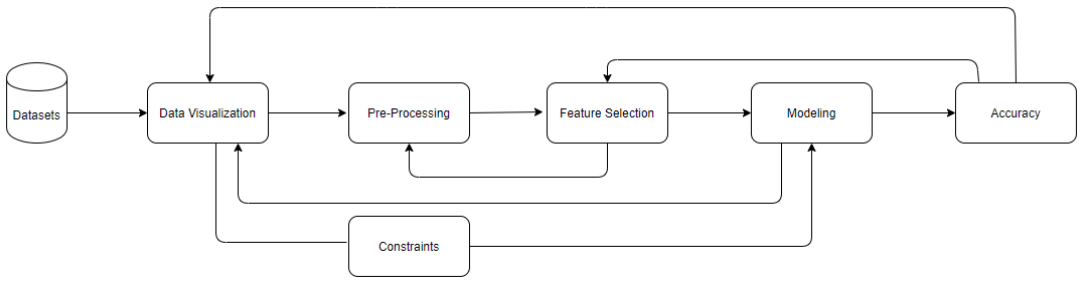


Figure 23. Experimental Setup Diagram

Description:

We selected a few of the most popular modelling architectures, visualised and pre-processed our dataset's data, after that we selected features on which we wanted to build our model. Then we visualized the accuracy of the model that we built and observed the constraints.

Instance: A single row of data is called an instance.

Datasets is a collection of instances is a dataset and when working with machine learning methods we typically need a few datasets for different purposes.

Training Dataset is a dataset that we feed into our machine learning algorithm to train our model.

Data visualization is a technique that uses an array of static and interactive visuals within our specified context to help and make sense of large amounts of data.

Preprocessing refers to all the transformations on the raw data before it is fed to the machine learning or deep learning algorithm.

Feature selection, also known as variable selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction.
While building a deep learning **model** we specified three layer types:

- The input layer is the layer to which you'll pass the features of your dataset. There is no computation that occurs in this layer. It serves to pass features to the hidden layers.
- The hidden layers are usually the layers between the input layer and the output layer—and there can be more than one. These layers perform the computations and pass the information to the output layer.
- The output layer represents the layer of your neural network that gives you the results after training your model. It is responsible for producing the output variables.

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Imposing constraints on the output of a Deep Neural Net is one way to improve the quality of its predictions while loosening the requirements for labeled training data.

6.2 Results

In the case of Tuberculosis, the final model predicted the results with an accuracy of 94%. During stress we fed unseen data to the trained model and following curves were plotted to show various aspects of the model's behavior. Our developed model successfully predicted abnormal cases of tuberculosis with the same accuracy.

6.2.1 Confusion Matrices

Tuberculosis detection In Chest X-rays

Table 3. Confusion Matrices Comparison in tuberculosis

	Accuracy	**Sensitivity**	**Specificity**
ReLU	50	50	N/A
SoftMax	94	95.8	92.3
SoftPlus	96	96	96

Description: This table represents model's accuracy, sensitivity and specificity comparison in Tuberculosis detection in chest X-ray using different activation functions.

- **Confusion matrix (ReLU)**

```
Confusion matrix, without normalization
[[25  0]
 [25  0]]
```

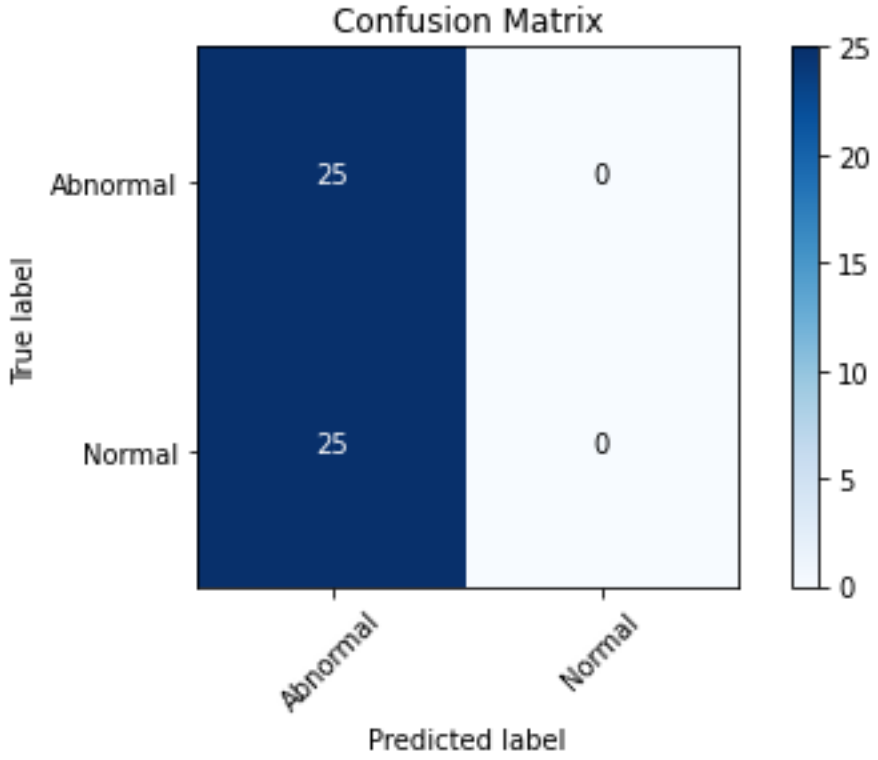


Figure 24. Confusion matrix (ReLU in tuberculosis)

Accuracy and Sensitivity were calculated by

Accuracy = (25+0)/(25+0+25+0) = 50
Sensitivity = 25/(25+25) = 50

- **Confusion matrix (SoftMax)**

```
Confusion matrix, without normalization
[[23  2]
 [ 1 24]]
```

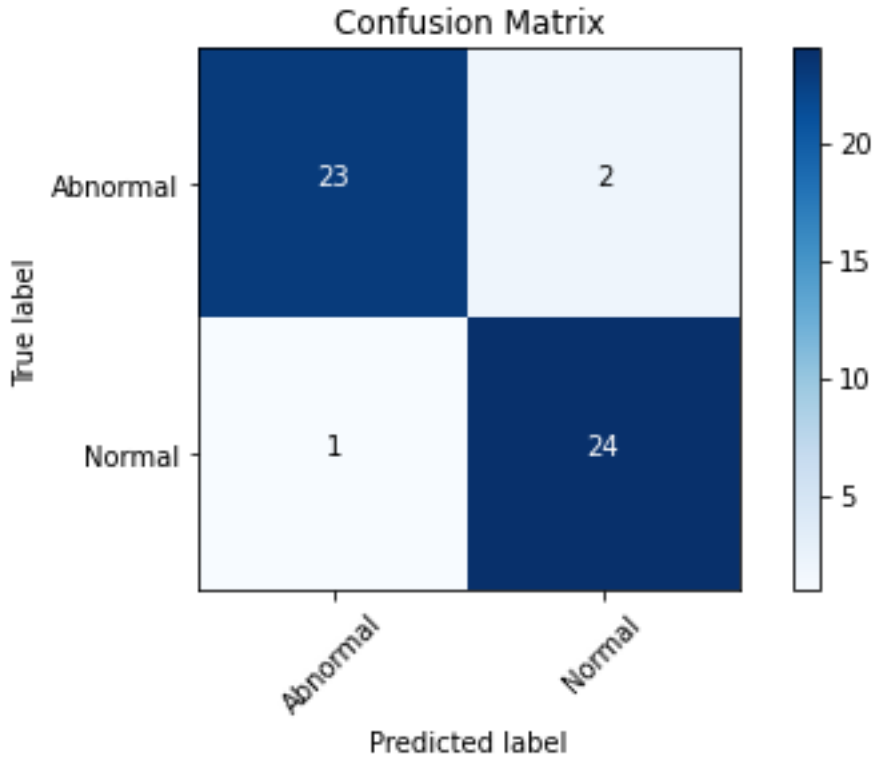


Figure 25. Confusion matrix (SoftMax in tuberculosis)

Accuracy, Specificity and Sensitivity were calculated by

Accuracy = (23+24)/(23+2+1+24) = 94
Sensitivity = 23/(23+1) = 95.8
Specificity = 24/(2+24) = 92.3

- **Confusion matrix (SoftPlus)**

```
Confusion matrix, without normalization
[[24  1]
 [ 1 24]]
```

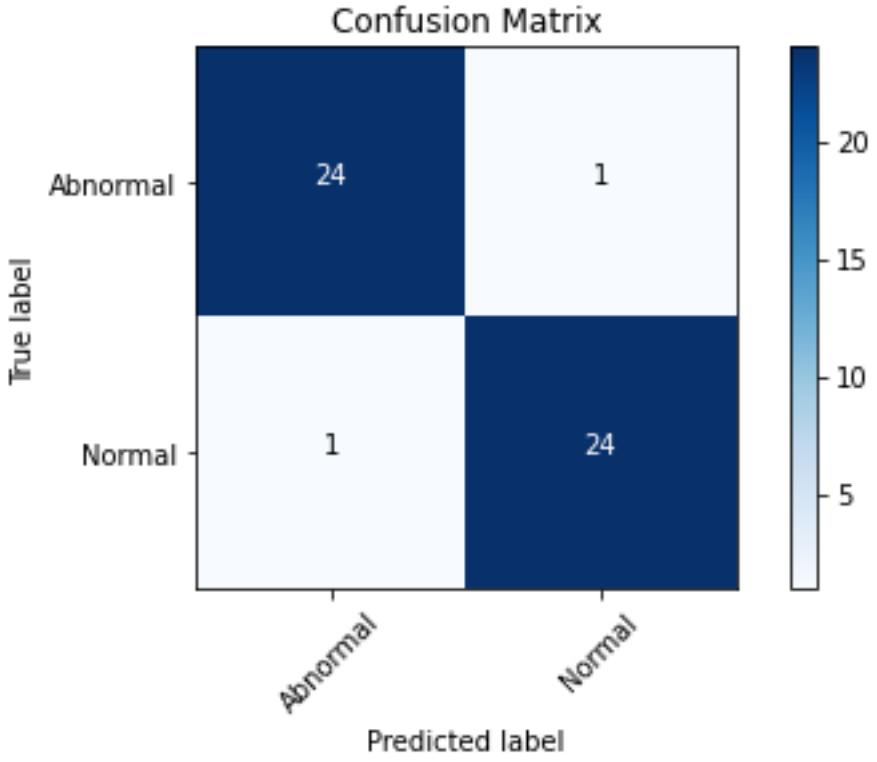


Figure 26. Confusion matrix (SoftPlus in tuberculosis)

Accuracy, Specificity and Sensitivity were calculated by

Accuracy = (24+24)/(24+1+24+1) = 96

Sensitivity = 24/(24+1) = 96

Specificity = 24/(1+24) = 96

6.2.2 Cross Entropy Loss

Cross Entropy Loss Comparison Table:

Table 4. Cross Entropy Comparison in tuberculosis

	Value	Epochs
ReLU	1	30
SoftMax	0.2	30
SoftPlus	0.2	30

Description: This table represents model's loss and val_loss Value per thirty Epochs comparison in Tuberculosis detection in chest X-ray using different activation functions

- **Cross Entropy Loss(ReLU)**

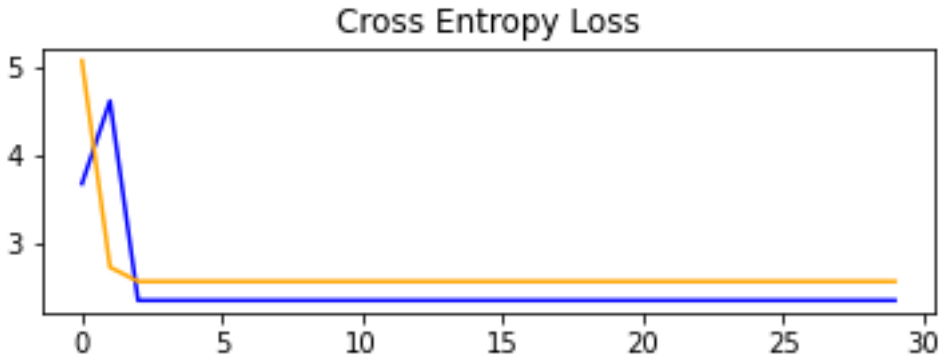


Figure 27. Cross Entropy Loss (Re LU in tuberculosis)

Shows Cross Entropy Loss of Re LU in tuberculosis.

Labels:

- **Blue**: loss
- **Orange**: val_loss
- **Y-axis**: value
- **X-axis**: epochs

- **Cross Entropy Loss(SoftMax)**

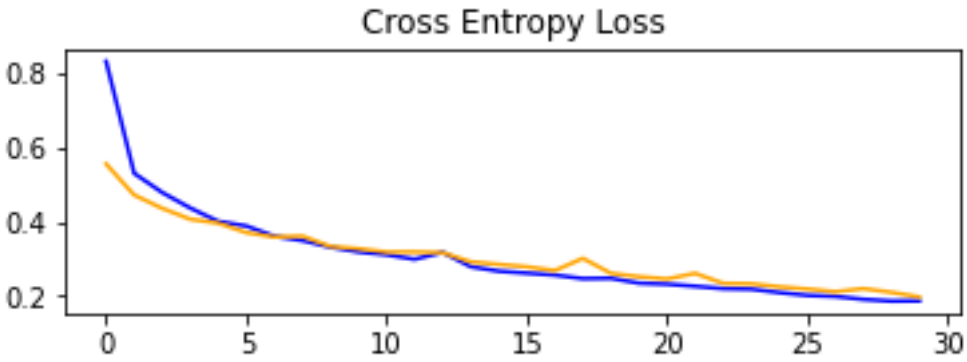


Figure 28. Cross Entropy Loss(SoftMax in tuberculosis)

Shows Cross Entropy Loss of SoftMax in tuberculosis

Labels:

- **Blue**: loss
- **Orange**: val_loss
- **Y-axis**: value
- **X-axis**: epochs

- **Cross Entropy Loss (SoftPlus)**

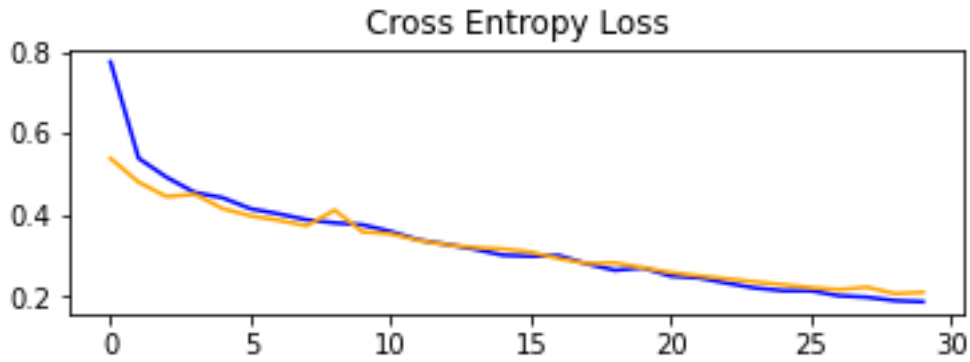


Figure 29. Cross Entropy Loss (Soft Plus in tuberculosis)

Description: Shows Cross Entropy Loss of Soft Plus in tuberculosis

Labels:

- **Blue**: loss
- **Orange**: val_loss
- **Y-axis**: value
- **X-axis**: epochs

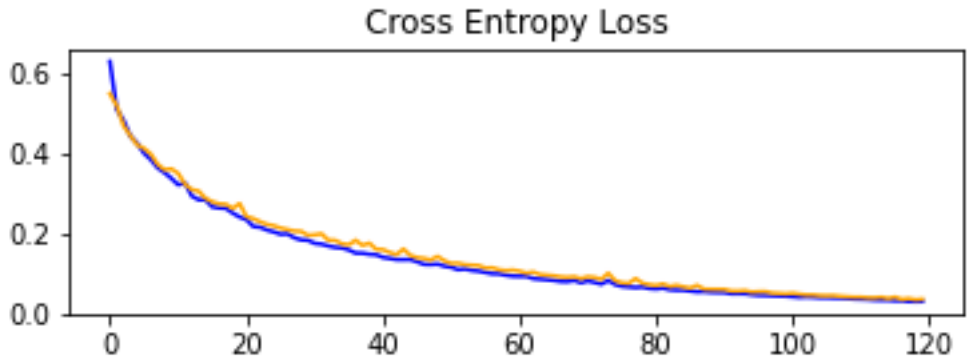


Figure 29.1 Cross Entropy Loss (SoftPlus in tuberculosis)

Shows Cross Entropy Loss of SoftPlus in tuberculosis

Labels:

- **Blue**: loss
- **Orange**: val_loss
- **Y-axis**: value
- **X-axis**: epochs

6.2.3 Classification Accuracy

Classification Accuracy Comparison Table:
Table 5. Classification Accuracy Comparison in tuberculosis

	Value	Epochs
ReLU	0.5	30
SoftMax	0.88	30
SoftPlus	0.94	30

This table represents model's accuracy and val_accuracy Value per thirty Epochs comparison in Tuberculosis detection in chest X-ray using different activation functions
Labels:

- **Blue**: accuracy
- **Orange**: val_accuracy
- **Y-axis**: value
- **Classification Accuracy (ReLU)**

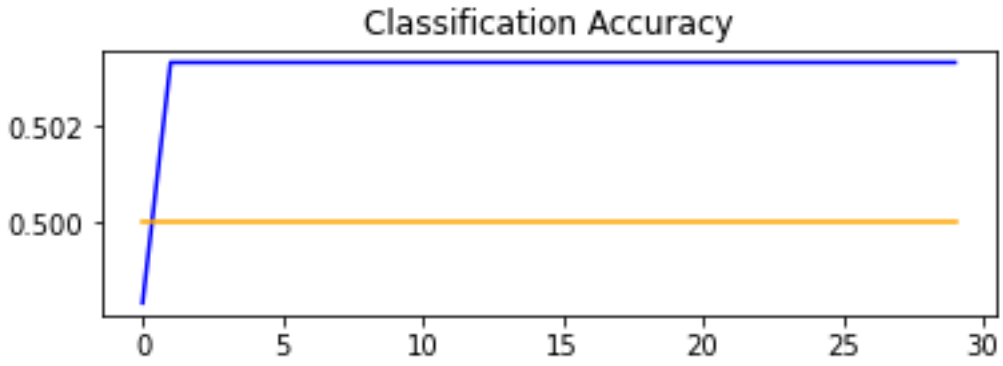


Figure 30. Classification Accuracy (ReLU in tuberculosis)

Shows Classification Accuracy of ReLU in tuberculosis

Labels:

- **Blue**: accuracy
- **Orange**: val_accuracy
- **Y-axis**: value
- **X-axis**: epochs

- **Classification Accuracy(SoftMax)**

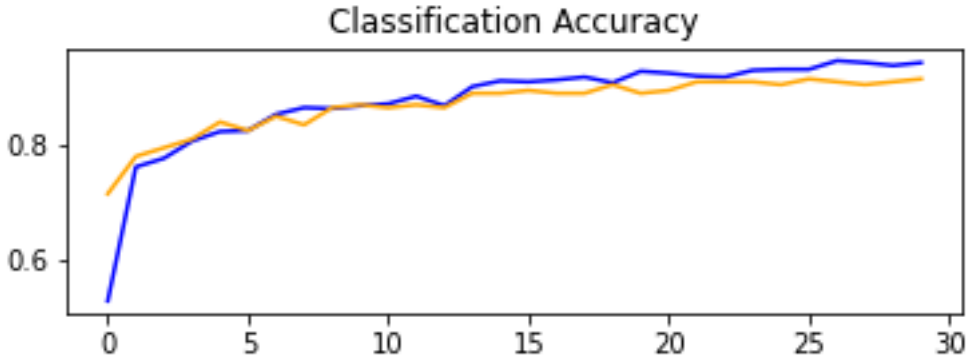


Figure 31. Classification Accuracy (SoftMax in tuberculosis)

Shows Classification Accuracy of SoftMax in tuberculosis

Labels:

- **Blue**: accuracy
- **Orange**: val_accuracy
- **Y-axis**: value
- **X-axis**: epoch
- **Classification Accuracy (SoftPlus)**

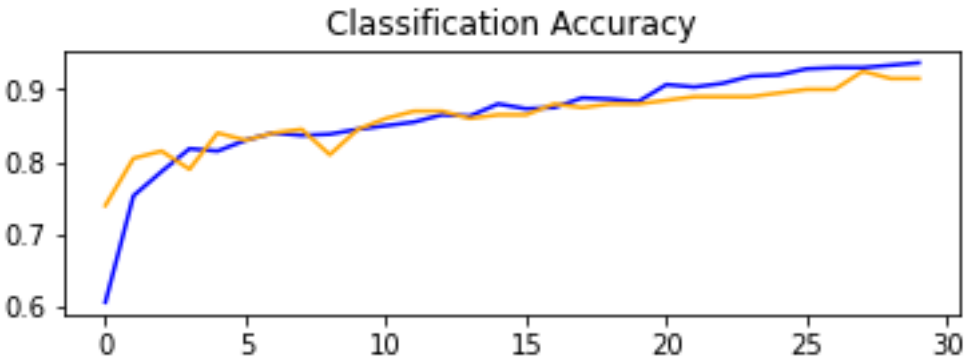


Figure 32. Classification Accuracy (Soft Plus in tuberculosis)

Shows Classification Accuracy of Soft Plus in tuberculosis

Labels:

- **Blue**: accuracy
- **Orange**: val_accuracy
- **Y-axis**: value
- **X-axis**: epochs

Breast Cancer Detection in Mammograms

6.2.4 Confusion matrix

Table 6. Confusion Matrix Comparison in Breast Cancer detection

	Accuracy	**Sensitivity**
ReLU	94.51	92.94
SoftMax	89.63	91.14
SoftPlus	85.37	82.2

Description: This table represents model's accuracy and sensitivity comparison in Tuberculosis detection in chest X-ray using different activation functions

- **Confusion matrix (ReLU)**

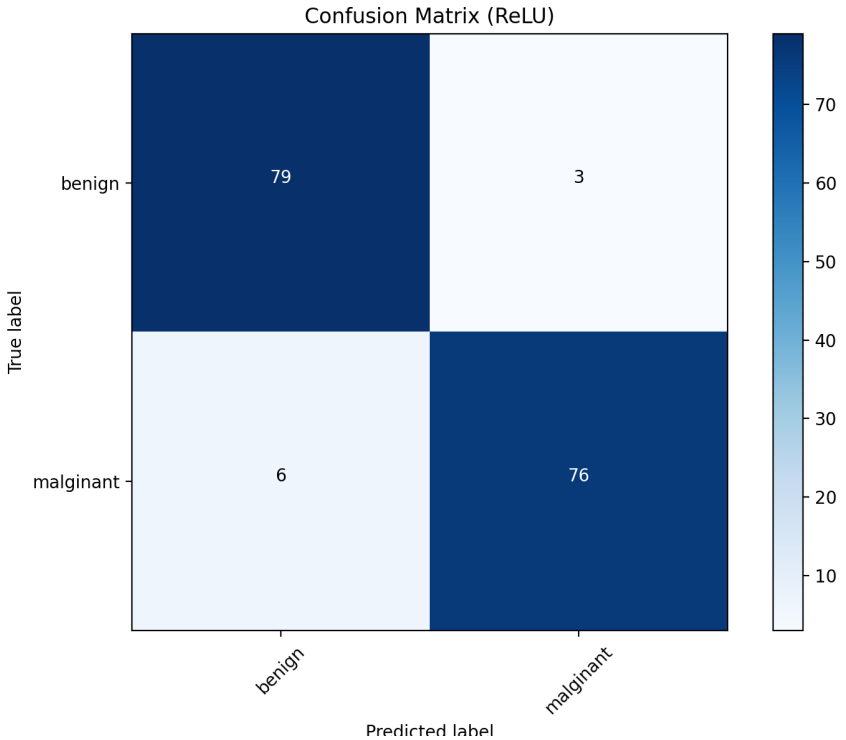


Figure 33. Confusion Matrix (ReLU in Breast Cancer Detection)

Accuracy and Sensitivity were calculated by

Accuracy = (79+76)/(79+3+6+70) = 94.51%
Sensitivity = 79/(79+6) = 92.94%

- **Confusion matrix (SoftMax)**

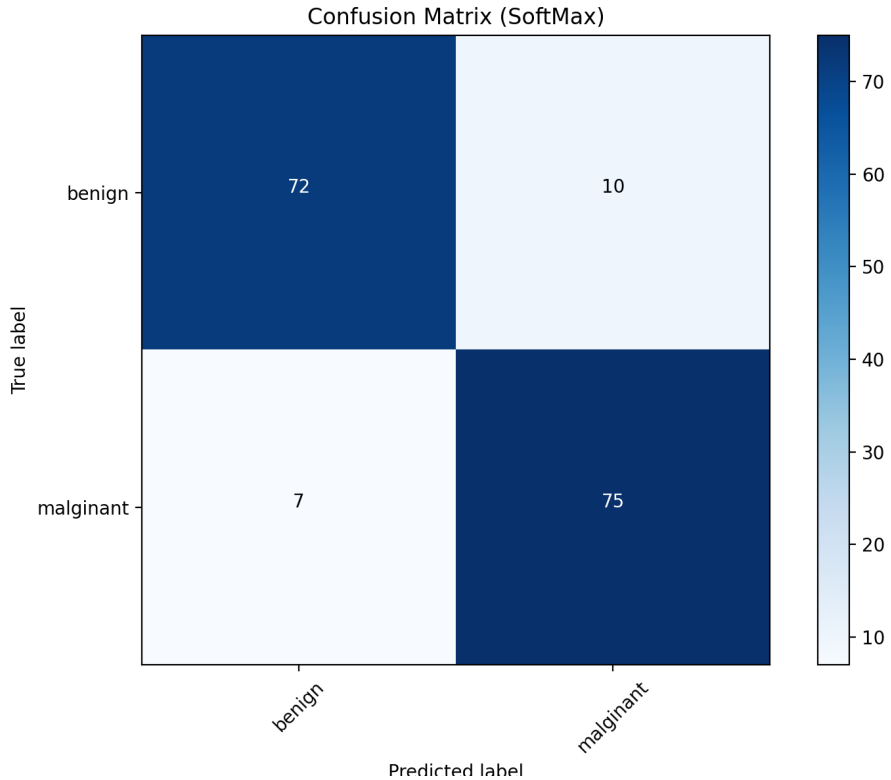


Figure 34. Confusion Matrix (SoftPlus in Breast Cancer Detection)

Accuracy and Sensitivity were calculated by

Accuracy = (72+75)/(72+10+7+75) = 89.63%
Sensitivity = 72/(72+7) = 91.14%

- **Confusion matrix (SoftPlus)**

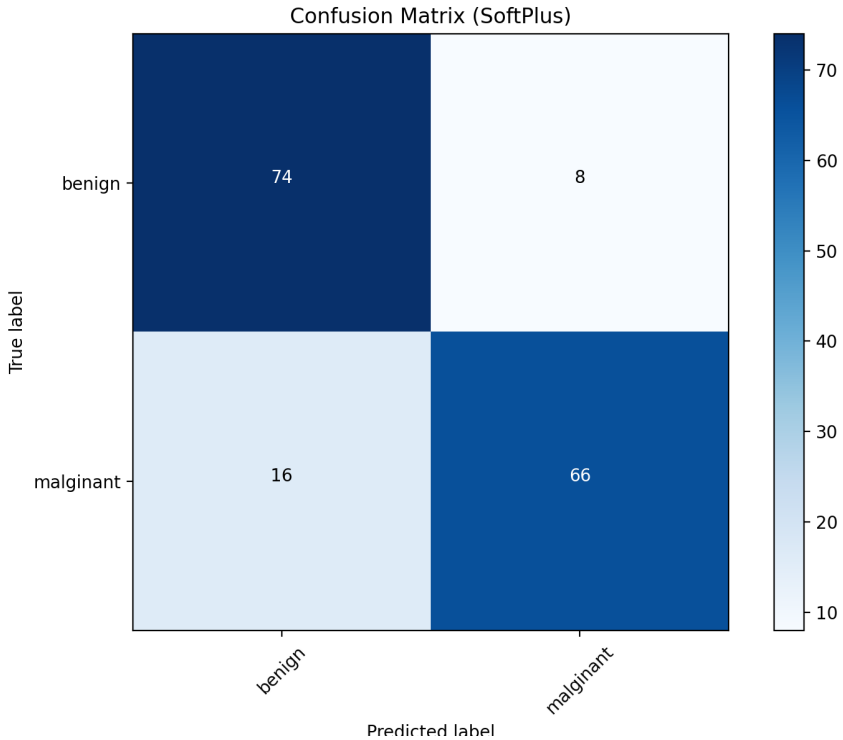


Figure 35. Confusion Matrix (SoftPlus in Breast Cancer Detection)

Accuracy and Sensitivity were calculated by

Accuracy = (74+66)/(74+8+16+66) = 85.37%
Sensitivity = 74/(74+16) = 82.2%

6.2.5 Cross Entropy Loss

Cross Entropy Loss Comparison Table:

Table 7. Cross Entropy Loss Comparison in Breast Cancer detection

	Value	Epochs
ReLU	0.2	80
SoftMax	0.2	80
SoftPlus	0.2	80

This table represents model's loss and val_loss Value per eighty Epochs.

- **Cross Entropy Loss (ReLU)**

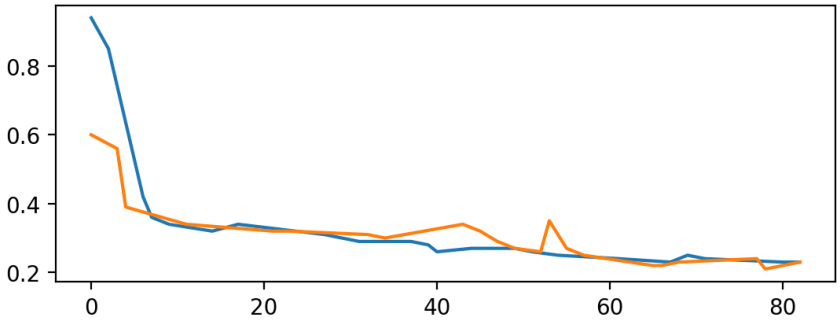


Figure 36. Cross Entropy Loss (ReLU in Breast Cancer Detection)

Shows Cross Entropy Loss of ReLU in Breast Cancer Detection

Labels:

- **Blue**: loss
- **Orange**: val_loss
- **Y-axis**: value
- **X-axis**: epochs

- **Cross Entropy Loss (SoftMax)**

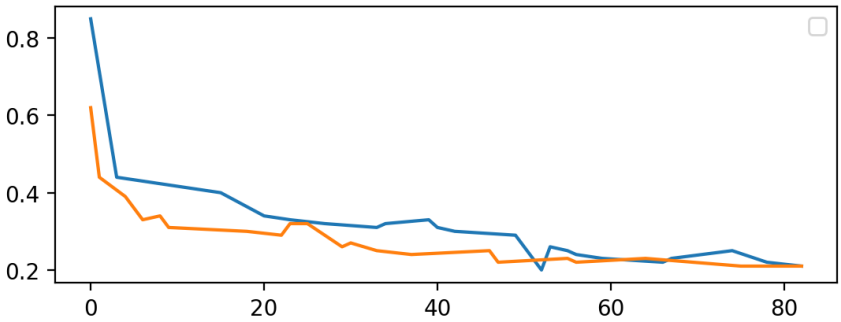


Figure 37. Cross Entropy Loss (SoftMax in Breast Cancer Detection)

Shows Cross Entropy Loss of SoftMax in Breast Cancer Detection

Labels:

- **Blue**: loss
- **Orange**: val_loss
- **Y-axis**: value
- **X-axis**: epochs

- **Cross Entropy Loss (SoftPlus)**

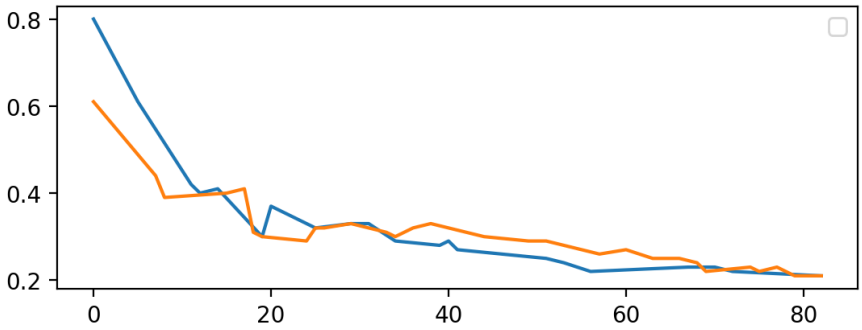


Figure 38. Cross Entropy Loss (SoftPlus in Breast Cancer Detection)

Shows Cross Entropy Loss of SoftPlus in Breast Cancer Detection

Labels:

- **Blue**: loss
- **Orange**: val_loss
- **Y-axis**: value
- **X-axis**: epochs

6.2.6 Classification Accuracy

Classification Accuracy Comparison Table:

Table 8. Classification Accuracy Comparison in Breast Cancer detection

	Value	Epochs
ReLU	0.88	80
SoftMax	0.92	80
SoftPlus	0.85	80

This table represents model's accuracy and val_accuracy Value per eighty Epochs comparison in Tuberculosis detection in chest X-ray using different activation functions

- **Classification Accuracy (ReLU)**

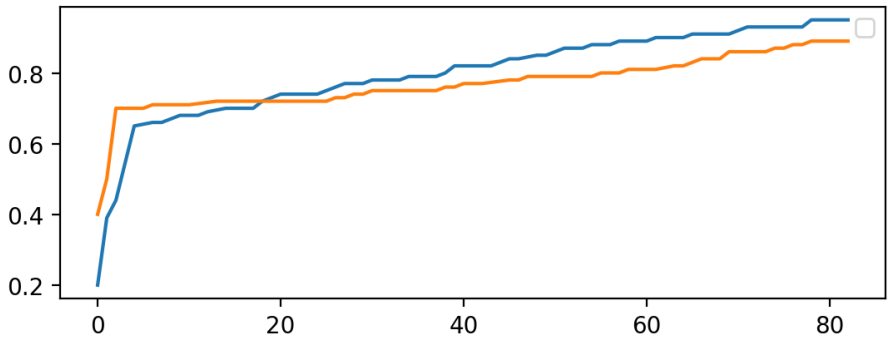


Figure 39. Classification Accuracy (ReLU in Breast Cancer Detection)

Shows Classification Accuracy of ReLU in Breast Cancer Detection

Labels:

- **Blue**: accuracy
- **Orange**: val_accuracy
- **Y-axis**: value
- **X-axis**: epochs

- **Classification Accuracy (SoftMax)**

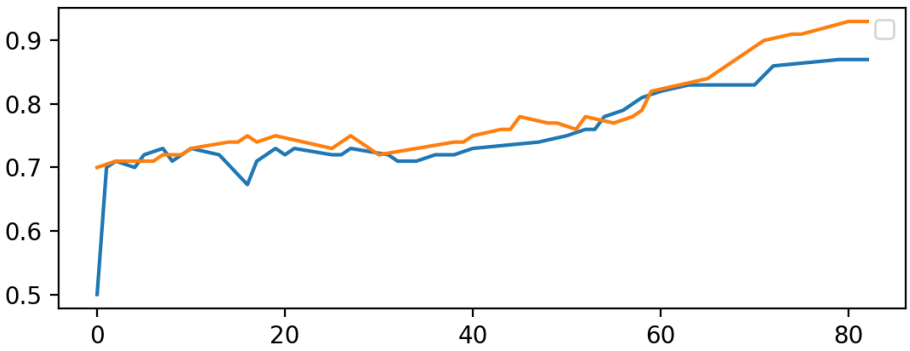


Figure 40. Classification Accuracy (SoftMax in Breast Cancer Detection)

Shows Classification Accuracy of SoftMax in Breast Cancer Detection

Labels:

- **Blue**: accuracy
- **Orange**: val_accuracy
- **Y-axis**: value
- **X-axis**: epochs

- **Classification Accuracy (SoftPlus)**

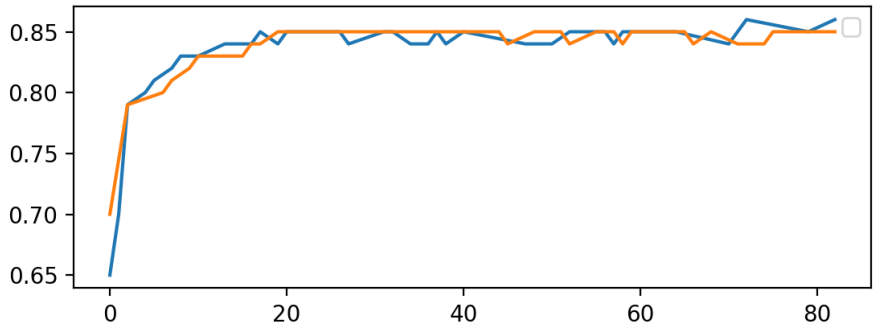


Figure 41. Classification Accuracy (SoftPlus in Breast Cancer Detection)

Shows Classification Accuracy of SoftPlus in Breast Cancer Detection

Labels:

- **Blue**: accuracy
- **Orange**: val_accuracy
- **Y-axis**: value
- **X-axis**: epochs

6.2.7 Cross Entropy Loss Comparison of both models

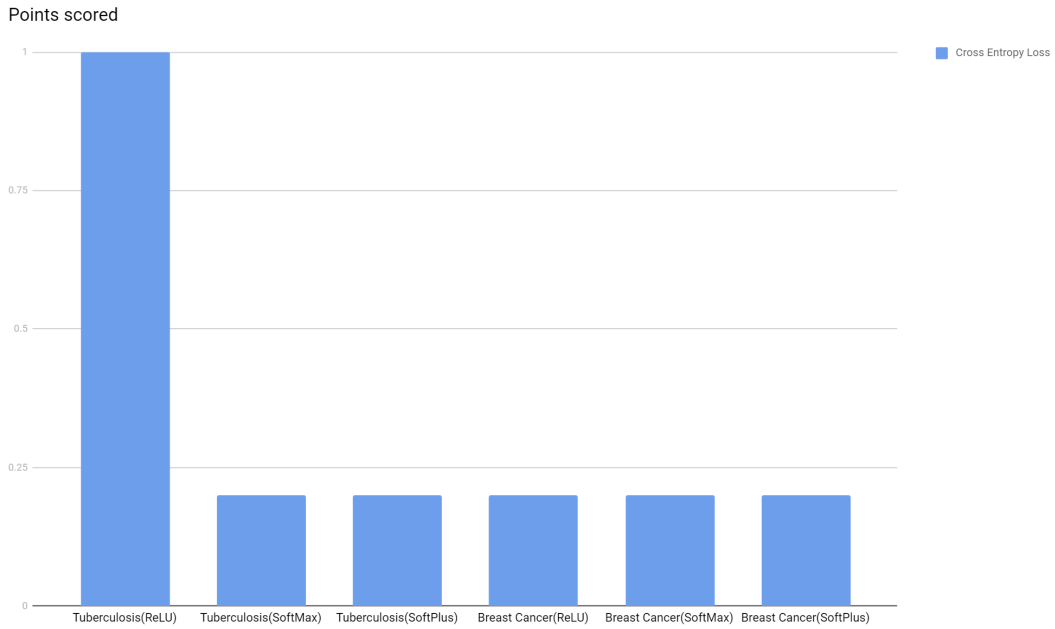


Figure 42. Cross Entropy Loss Comparison in Both models.

Shows Cross Entropy Loss Comparison in Both models.

6.2.8 Classification Accuracy Comparison of both models

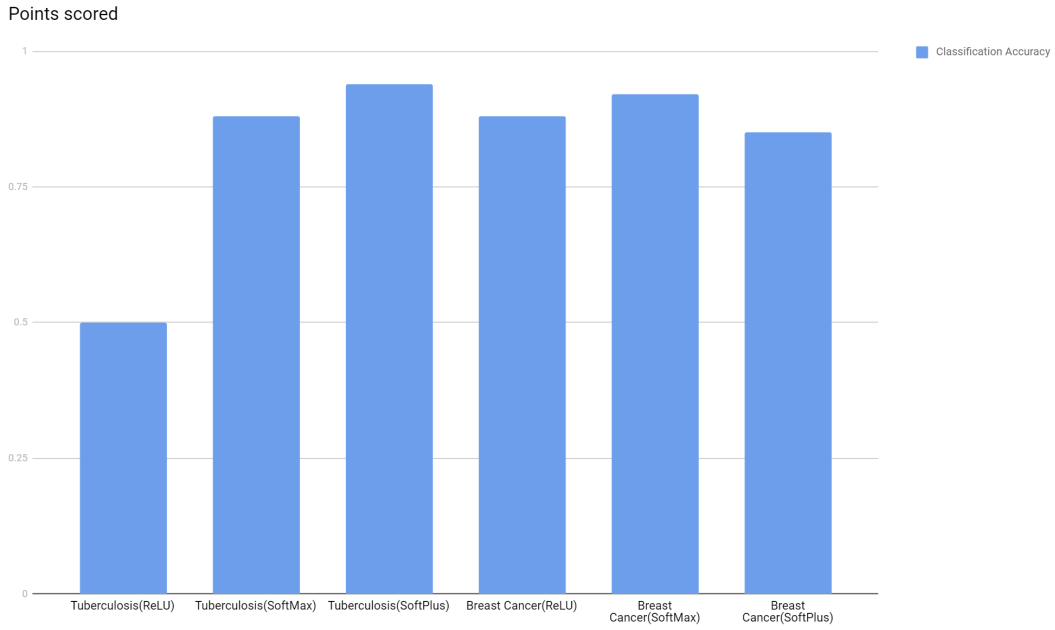


Figure 43. Classification Accuracy Comparison in Both models

Shows Classification Accuracy Comparison in Both models

6.2.9 Confusion Matrix Comparison of both models

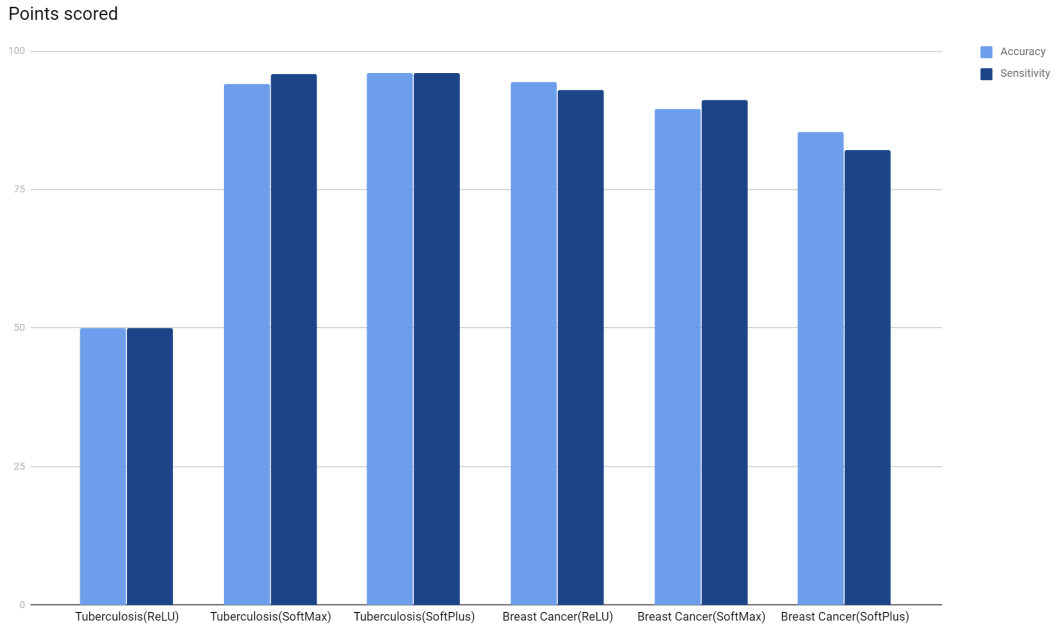


Figure 44. Confusion Matrix Comparison in Both models

Shows Confusion Matrix Comparison in Both models

6.3 Discussion/Analysis

The main two problems faced during this whole process are discussed below.

6.3.1 Model Constraints

One goal was to choose a model which had least constraints on data so the information is not lost in the process. Our model selection technique worked well as there are only three constraints:

Selecting data in pictorial form.

Choosing datasets with image extension '.png' to preserve image quality.

Selecting valid and large enough datasets for model training to achieve good accuracy.

Solution:

We applied filters that only allow image data to be passed.

The filters applied also checked if the input (image file) being passed has the acceptable file extension that works best with the model

We selected the China data set (Database), which is the better digital image database in screening the Tuberculosis. It was created by National Library of Medicine, Maryland, USA with the collaboration of Guangdong Medical College, Shenzhen, China. The data was gathered from outside the clinics or the normal routine patient.

6.3.2 Dataset Limitation

In the case of Tuberculosis, one limitation of the model is due to the dataset limitation as it is trained on one dataset. It can accurately predict the X-rays which are similar to the dataset, model is trained on.

In the case of Breast Cancer, the major limitation is providing the laterality and view of the mammogram as input to the process. Since this involves technical knowledge, the accuracy of the prediction might get skewed if an incorrect input is provided. While this model also lacks data from different sets of populaces. We will spread our network in the future with better model training.

A large scale, breast cancer mammogram and tuberculosis chest X-ray database will help to create more generalized models which will be more accurate.

6.3.3 Comparison with similar projects

Table 9. Accuracy of Model in comparison to other similar models

	Technique	Dataset	Pre-Processing	Accuracy
Seelwan Sathitratanacheewin, Panasun Sunanta, Krit Pongpirul (2020)	Inception V3	National Library of Medicine Shenzhen No.3 Hospital	color-space, crop-flip, rotational methods	70.54%
Tae Kyung Kim, Paul H. Yi, Gregory D. Hager, Cheng Ting Lin1 (2019)	DCNN	Shenzhen China set	Heatmaps	91%
Proposed Work	VGG16 using SoftPlus activation function	China Set - The Shenzhen set - Chest	keras.applications.vgg16.preprocess_input	94%

This table represents model's accuracy in comparison to other similar models.

7 CONCLUSION AND FUTURE WORK

In conclusion, our research shows that deep learning models can accurately classify breast cancer or tuberculosis. The reason we chose these two diseases was to demonstrate that an artificial neural network has the capability to predict most common diseases nowadays. The advancement of computer technology has played a huge part in our success. It would be impossible to train this model in a completely end-to-end fashion with currently available hardware. Although our results are promising, we admit that the test set used in our experiments is relatively small and our results require further clinical substantiation.

When it comes to detection of a certain disease, screening is only the first process, with a certain medical professional deciding the final verdict. The whole idea behind this research lies in the fact that with the aid of prediction made by artificial intelligence, medical practitioners and even the patient themselves can have easy access to firsthand knowledge about the disease. With this technology, we can streamline the whole process of detection and save more lives by detecting illness at it's earliest.

In addition to these models, the next step would be not only improving the ones we have already created but also add more models, specifically for other diseases. This is the potential future of medical diagnosis, predicting the development of illness before it is even catchable by humans. Therefore, we hope to get funding and collaboration from investors and scientists around the world in building and evolving the IRMI system. We have successfully rolled out APIs for Breast Cancer Detection and Tuberculosis Detection. These can be utilized by any device which is connected to the internet.

8 REFERENCES/ BIBLIOGRAPHY

1. Lodwick GS, Keats TE, Dorst JP. The coding of Roentgen images for computer analysis as applied to lung cancer. Radiology. 1963

2. Zakirov AN, Kuleev RF, Timoshenko AS, Vladimirov AV. Advanced approaches to computer-aided detection of thoracic diseases on chest X-rays. Appl Math Sci. 2015 http://www.m-hikari.com/ams/ams-2015/ams-85-88-2015/54348.html .

3. van Ginneken B, Hogeweg L, Prokop M. Computer-aided diagnosis in chest radiography: beyond nodules. Eur J Radiol. 2009

4. Betsy Antony, Nizar Banu P K. Lung Tuberculosis Detection Using X-Ray Images, 2017

5. Pendharkar P, Rodger J, Yaverbaum G, Herman N, Benner M. Association, statistical, mathematical and neural approaches for mining breast cancer patterns. Expert Syst Appl. 1999

6. Danenas P, Garsva G. Selection of Support Vector Machines based classifiers for credit risk domain. Expert Syst Appl. 2015

7. VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION by Karen Simonyan & Andrew Zisserman

8. A. M. Kutay, A. P. Peropulu, and C. Piccoli, **"**Tissue characterization based on the power-law shot noise model**"** Pattern Recogn. Lett., 2002.

9. Guardado-Medina R. O., Lepe R. R. and Magana B. O. **"**Intelligent detection of microcalcifications in mammograms based on fuzzy techniques**"** 2012.

10. Malkov S., Shepherd J. A., Scott C. G., Tamimi R. M., Ma L., Bertrand K. A. and Norman A. "Mammographic texture and risk of breast cancer by tumor type and estrogen receptor status Breast Cancer Research" 2016.

11. Aswathy M. A. and Jagannath M. 2017 "Detection of breast cancer on digital histopathology images" Present status and future possibilities Informatics in Medicine.

12. Ratula Ray1, Azian Azamimi Abdullah2, Debasish Kumar Mallick3 and Satya Ranjan DashClassification of Benign and Malignant Breast Cancer using Supervised Machine Learning Algorithms Based on Image and Numeric Datasets.

13. B. Caputo, T. Tommasi, H. Müller, T. M. Deserno and J. Kalpathy-Cramer, "ImageCLEF 2009 lung nodule detection and medical annotation task", 2009, [online] Available: http://www.imageclef.org/2009/medanno.

14. Automatic screening for tuberculosis in chest radiographs: a survey", *Quantitative imaging in medicine and surgery*, 2013.

15. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun "Deep Residual Learning for Image Recognition".

16. Manoj Raju Arun Aswath Venkatesh Pagidimarri "Automatic Detection for Tuberclosis Using Deep Learning Methods".

17. J. Burrill, C.J. Williams, G. Bain, G. Conder, A.L. Hine, and R.R. Misra. "Tuberculosis: A Radiologic Review". RadioGraphics, 2007, Vol. 27, No. 5, 1255-1273.

18. S. Jaeger, A. Karargyris, S. Candemir, L. Folio, J. Siegelman, F. Callaghan, Z. Xue, K. Palaniappan, R.K. Singh, S. Antani, G. Thoma, Y.X. Wang, P.X. Lu, C.J. McDonald. Automatic tuberculosis screening using chest radiographs.

IEEE Trans Med Imaging. 2014 Feb;33(2):233-45. doi: 10.1109/TMI.2013.2284099. PMID: 24108713

19. S. K. Zhou, H. Greenspan and D. Shen, Deep learning for medical image analysis, Academic Press, 2017.

20. Y. Cao, C. Liu, B. Liu, M. J. Brunette, N. Zhang et al., "Improving tuberculosis diagnostics using deep learning and mobile health technologies among resource-poor and marginalized communities", *Connected Health: Applications Systems and Engineering Technologies (CHASE). IEEE*, 2016

10 APPENDIX

10.1 Source Documents

Tuberculosis model creation and results:

In [1]:
```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, Flatten, BatchNormalization, Conv2D, MaxPool2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix
import itertools
import os
import shutil
import random
import glob
import matplotlib.pyplot as plt
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
%matplotlib inline
```

In [2]:
```
physical_devices = tf.config.experimental.list_physical_devices('GPU')
print("Num GPUs Available: ", len(physical_devices))
tf.config.experimental.set_memory_growth(physical_devices[0], True)
```

```
Num GPUs Available:  1
```

In [4]:
```
train_path = 'TB/TB/train'
valid_path = 'TB/TB/valid'
test_path = 'TB/TB/test'
```

In [5]:
```
train_batches = ImageDataGenerator().flow_from_directory(directory=train_path, target_size=(224,224), classes=['abnormal', 'norma
valid_batches = ImageDataGenerator().flow_from_directory(directory=valid_path, target_size=(224,224), classes=['abnormal', 'norma
test_batches = ImageDataGenerator().flow_from_directory(directory=test_path, target_size=(224,224), classes=['abnormal', 'normal'
```

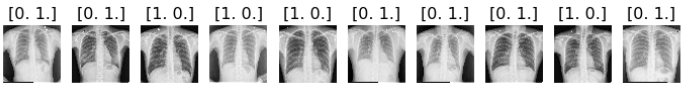
```
Found 600 images belonging to 2 classes.
Found 200 images belonging to 2 classes.
Found 50 images belonging to 2 classes.
```

In [6]:
```
assert train_batches.n == 600
assert valid_batches.n == 200
assert test_batches.n == 50
assert train_batches.num_classes == valid_batches.num_classes == test_batches.num_classes == 2
```

In [7]:
```
imgs, labels = next(train_batches)
```

In [8]:
```
# plots images with labels within jupyter notebook
def plots(ims, figsize=(12,6), rows=1, interp=False, titles=None):
    if type(ims[0]) is np.ndarray:
        ims = np.array(ims).astype(np.uint8)
        if (ims.shape[-1] != 3):
            ims = ims.transpose((0,2,3,1))
    f = plt.figure(figsize=figsize)
    cols = len(ims)//rows if len(ims) % 2 == 0 else len(ims)//rows + 1
    for i in range(len(ims)):
        sp = f.add_subplot(rows, cols, i+1)
        sp.axis('Off')
        if titles is not None:
            sp.set_title(titles[i], fontsize=16)
        plt.imshow(ims[i], interpolation=None if interp else 'none')
```

In [9]:
```
plots(imgs, titles = labels)
```



In [10]:
```
vgg16_model = tf.keras.applications.vgg16.VGG16()
```

In [11]:
```
vgg16_model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
_________________________________________________________________
```

In [12]:
```
type(vgg16_model)
```

Out[12]:
```
tensorflow.python.keras.engine.training.Model
```

In [13]:
```
model = Sequential()
for layer in vgg16_model.layers[:-1]:
    model.add(layer)
```

In [14]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
=================================================================
Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0
_________________________________________________________________
```

In [15]:
```
for layer in model.layers:
    layer.trainable = False
```

In [16]:
```
model.add(Dense(units=2, activation='softplus'))
```

In [17]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
dense (Dense)                (None, 2)                 8194
=================================================================
Total params: 134,268,738
Trainable params: 8,194
Non-trainable params: 134,260,544
_________________________________________________________________
```

In [18]:
```
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```

In [19]:
```
model.fit(x=train_batches,
          steps_per_epoch=len(train_batches),
          validation_data=valid_batches,
          validation_steps=len(valid_batches),
          epochs=300,
          verbose=2
)
```

```
Epoch 1/300
60/60 - 65s - loss: 0.6095 - accuracy: 0.6850 - val_loss: 0.5220 - val_accuracy: 0.7900
Epoch 2/300
60/60 - 64s - loss: 0.4876 - accuracy: 0.7933 - val_loss: 0.4660 - val_accuracy: 0.7900
Epoch 3/300
60/60 - 65s - loss: 0.4558 - accuracy: 0.8117 - val_loss: 0.4372 - val_accuracy: 0.8200
Epoch 4/300
60/60 - 66s - loss: 0.4402 - accuracy: 0.8067 - val_loss: 0.4425 - val_accuracy: 0.8200
Epoch 5/300
60/60 - 65s - loss: 0.4071 - accuracy: 0.8400 - val_loss: 0.3883 - val_accuracy: 0.8400
Epoch 6/300
60/60 - 70s - loss: 0.3847 - accuracy: 0.8450 - val_loss: 0.3712 - val_accuracy: 0.8600
Epoch 7/300
60/60 - 69s - loss: 0.3700 - accuracy: 0.8517 - val_loss: 0.3589 - val_accuracy: 0.8500
Epoch 8/300
60/60 - 67s - loss: 0.3489 - accuracy: 0.8617 - val_loss: 0.3456 - val_accuracy: 0.8650
Epoch 9/300
60/60 - 67s - loss: 0.3374 - accuracy: 0.8767 - val_loss: 0.3430 - val_accuracy: 0.8500
Epoch 10/300
```

In [18]:
```
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```

In [19]:
```
model.fit(x=train_batches,
          steps_per_epoch=len(train_batches),
          validation_data=valid_batches,
          validation_steps=len(valid_batches),
          epochs=300,
          verbose=2
)
```

```
60/60 - 60s - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.0016 - val_accuracy: 1.0000
Epoch 293/300
60/60 - 60s - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.0016 - val_accuracy: 1.0000
Epoch 294/300
60/60 - 60s - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.0016 - val_accuracy: 1.0000
Epoch 295/300
60/60 - 60s - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.0018 - val_accuracy: 1.0000
Epoch 296/300
60/60 - 60s - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.0015 - val_accuracy: 1.0000
Epoch 297/300
60/60 - 60s - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.0015 - val_accuracy: 1.0000
Epoch 298/300
60/60 - 60s - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0014 - val_accuracy: 1.0000
Epoch 299/300
60/60 - 60s - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0015 - val_accuracy: 1.0000
Epoch 300/300
60/60 - 60s - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0014 - val_accuracy: 1.0000
```

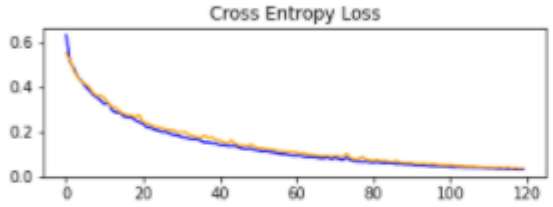
Out[19]:
```
<tensorflow.python.keras.callbacks.History at 0x20e4c437278>
```

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

```
In [20]: predictions = model.predict(x=test_batches, steps=len(test_batches), verbose=0)

In [21]: test_batches.classes

Out[21]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1])

In [22]: cm = confusion_matrix(y_true=test_batches.classes, y_pred=np.argmax(predictions, axis=-1))

In [23]: test_batches.class_indices

Out[23]: {'abnormal': 0, 'normal': 1}
```

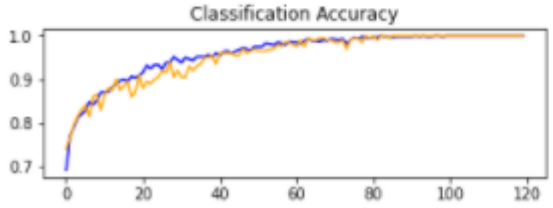
```
plt.subplot(211)
plt.title('Cross Entropy Loss')
plt.plot(history.history['loss'], color='blue', label='train')
plt.plot(history.history['val_loss'], color='orange', label='test')
```

```
[<matplotlib.lines.Line2D at 0x22cf67caa20>]
```




```
# plot accuracy
plt.subplot(212)
plt.title('Classification Accuracy')
plt.plot(history.history['accuracy'], color='blue', label='train')
plt.plot(history.history['val_accuracy'], color='orange', label='test')
```

```
[<matplotlib.lines.Line2D at 0x22d44718748>]
```




```
cm_plot_labels = ['Abnormal','Normal']
plot_confusion_matrix(cm=cm, classes=cm_plot_labels, title='Confusion Matrix')
```