IMPACT OF BONE SUPPRESSION IN TUBERCULOSIS DETECTION

A PROJECT REPORT

Submitted by

UTKARSH SINHA [Reg No: RA1911003010521]

TANYA KUMAR [Reg No: RA1911003010546]

Under the guidance of

Dr. C. Vijayakumaran

Associate Professor, Department of Computing Technologies

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGIES

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603203

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR–603 203

BONAFIDE CERTIFICATE

Certified that 18CSP109L / I8CSP111L project report titled "**Impact of Bone Suppression on Tuberculosis Detection**" is the bonafide work of **UTKARSH SINHA [RegNo:RA1911003010521]** and **TANYA KUMAR [RegNo:RA1911003010546]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr. C. VIJAYAKUMARAN
SUPERVISOR
Associate Professor
Department of Computing Technologies

Dr. C. VIJAYAKUMARAN
PANEL HEAD
Associate Professor
Department of Computing Technologies

Dr. M. PUSHPALATHA
HEAD OF THE DEPARTMENT
Department of Computing Technologies

INTERNAL EXAMINER

EXTERNAL EXAMINER

Department of Computing Technologies
SRM Institute of Science and Technology
Own Work Declaration Form

Degree/Course: B.Tech in Computer Science and Engineering

Student Names: UTKARSH SINHA , TANYA KUMAR

Registration Number: RA1911003010521, RA1911003010546

Title of Work: Impact of Bone suppression on Tuberculosis Detection

I/We here by certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text(from books, web,etc.)
- Given the sources of all pictures, data etc that are not my own.
- Not made any use of the report(s) or essay(s) of any other student(s)either past or present
- Acknowledged in appropriate places any help that I have received from others(e.g fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course hand book / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above. **Student 1 Signature:** **Student 2 Signature:** **Date:**
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr. T. V. Gopal**, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, **Dr. M. Pushpalatha,** Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinators, **Dr. M. Kanchana**, **Dr. G. Usha, Dr. R. Yamini and Dr. K. Geetha**, Panel Head, **Dr. C. Vijayakumaran**, AssociateProfessor and Panel Members, **Dr. R. Jeya,** Assistant Professor, **Dr. J. Ramkumar,** Assistant Professor, and **Dr. J. Kalaivani,** Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. R. Yamini**, Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr. C. Vijayakumaran**, Associate Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his mentorship. He provided us with the freedom and support to explore the research topics of our interest. His passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff and students of the Computing Technologies Department, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement.

UTKARSH SINHA [Reg. No: RA1911003010521]

TANYA KUMAR [Reg. No: RA1911003010546]

ABSTRACT

Computed Tomography (CT) scans can provide high-quality images of the lungs without the obstruction of bones, making them useful for the diagnosis and monitoring of lung diseases such as Tuberculosis (TB) and lung cancer. However, CT scans have some drawbacks, including their time-consuming nature and the high radiation doses associated with their use. Due to these limitations, CT scans are generally not suitable for mass screening or routine diagnostic purposes. Instead, chest X-rays are often used as a first-line diagnostic tool for the detectionof lung diseases. Chest X-rays are inexpensive, widely available, and require simple equipment, making them a practical choice for large-scale screening programs. However, conventional chest X-rays may not always provide the necessary level of detail for accurate diagnosis, especially in cases where bone tissue obscures the view of the lungs. Dual Energy Subtraction (DES) is a technique that can be used toobtain clearer images of the lungs by removing the bone tissue from the X-ray image. However, this technique involves exposing the patient to X-ray radiation twice, which can increase the risk of radiation exposure. In conclusion, while CT scans can provide fine images of the lungs without bones, their use is limited due to their time-consuming nature and high radiation doses. Chest X-rays are a practical choice for mass screening and routine diagnostic purposes, but may require additional techniques such as DES for greater accuracy. Medical professionalsshould carefully evaluate the risks and benefits of different diagnostic techniques and tailor their approach to the individual needs of each patient.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

BS	Bone Suppression
TB	Tuberculosis
CNF	Convolutional Filter
CNN	Convolutional Neural Network
CT	Computed Tomography
AI	Artificial Intelligence
ML	Machine Learning
CXR	Chest X-Ray
AE	Autoencoder
CLS	Classification
DES	Dual Energy Spectroscopy
MTANN	Multi-Task Artificial Intelligence
GAN	Generative Adversarial Networks
ResNet	Residual Network
COPD	Chronic Obstructive Pulmonary Disease
i.e.	id est
e.g.	exempli gratia
Eq	Equation
MAE	Mean Absolute Error
MSE	Mean Squared Error

CHAPTER 1

INTRODUCTION

1.1 ARTIFICIAL INTELLIGENCE

While Machine Learning, Deep Learning, and Artificial Intelligence are all related concepts, they all refer to distinct fields of research and application.

AI is a vast field that includes any method or tool that enables computers to carry out operations like speech recognition, image recognition, decision-making, and natural language processing, which often need intelligence resembling that of a person.

In Machine Learning, a branch of Artificial Intelligence, computers are trained to automatically learn from their experiences and advance. It makes predictions or conclusions based on patterns in the data that are found using statistical techniques. supervised, unsupervised, or semi-supervised ML algorithms are all possible. A branch of machine learning called DL involves teaching multiple- layered artificial neural networks to learn from vast volumes of data. The neural networks can learnand get better over time since they are modelled after the human brain. Applications for DL algorithmsinclude speech and picture recognition, natural language processing, and predictive modelling.

ML and DL are subfields that concentrate on certain techniques and approaches for enabling machines to learn and make decisions, although AI is the overall most comprehensive field.

1.2 BONE - SUPPRESSION

Bone suppression is a technique used in medical imaging to enhance the visibility of soft tissues and abnormalities in areas that are overlaid by bones. In radiography, X-rays are absorbed more by dense materials such as bones than by softer tissues such as muscles and organs, resulting in images that are difficult to interpret in areas where bones and soft tissues overlap.

To address this issue, bone suppression algorithms can be applied to X-ray images to remove or reduce the appearance of bones while preserving the soft tissue information. One common approach is to use a Convolutional Neural Network (CNN) to learn a filter that separates the bone and soft tissue components of an X-ray image. The filter is then applied to the image, resulting in a bone-suppressed image that highlights the soft tissue structures of interest. Bone suppression can be useful in many medical applications, including chest X-rays for lung cancer screening and mammography for breast cancer detection. It can help radiologists and physicians more accurately identify and diagnose abnormalities, leading to earlier detection and better patient outcomes.

machines to learn and make decisions, although AI is the overall most comprehensive field.

1.3 OBJECTIVE

Chest x-ray and CT scan are two different imaging modalities used to diagnose chest diseases. A chest x-ray is a non-invasive diagnostic imaging test that produces images of the lungs, heart, blood vessels and surrounding tissues, using small amounts of radiation. It provides a two-dimensional image of the chest, making it an excellent tool for screening for lung cancer, pneumonia, tuberculosis, and other respiratory conditions. It is also quick, affordable, and widely available. On the other hand, CT scan is a more advanced imaging test that uses X-rays and computer technology to generate three-dimensional images of the chest. It is excellent in detecting small abnormalities and is often used for detailed

evaluation of lung and mediastinal structures.

CT scan may be appropriate for diagnosing conditions such as lung cancer, pulmonary embolism, and pulmonary fibrosis. While CT scan provides more detailed images, it also involves more radiation exposure and is more expensive. Therefore, the choice of imaging depends on the clinical situation and the specific needs of the patient**.** CT Scans can be used for fine images without bones however they are time consuming and have large radiation doses hence are not suitable for mass screening.

X-Rays are inexpensive and require simple equipment. For example, a CT scan delivers 7mSv(sievert) which is roughly 70 times that of an X-ray. The other method of getting a Chest Xray without the obstruction of bones is DES which involves the patient being exposed to the X-Ray radiations twice. For better and earlier mass diagnosis/screening of diseases such as Tuberculosis, we need a model with good accuracy to generate bone-suppressed images, images that suppress bones and enhance soft tissues. However, to really understand whether such a model is effective, we do a comparison of the bone suppressed images and normal x-rays and see which gives higher accuracy for tuberculosis classification and quicker diagnosis.

1.4 AIM

- Firstly, we will create an autoencoder model that outputs Chest X-Ray (CXR) images with bone suppression.
- Next, we will create a classification model for Tuberculosis using normal x-ray images
- After that, we will create a classification model for the bone suppressed images and check for performance differences between the two models.
- Finally, we will note down the benefits and drawbacks of the resulting diagnosis compared to standard chest x-rays.

1.5 SCOPE

There can be various implementations of our deep learning model , in numerous real world applications. While we focus on the detection of a particular disease, the model can be furtherenhanced to detect other lung diseases such as Pneumonia, lung cancer, Tuberculosis etc. Chest detection models can also be integrated into computer-aided diagnosis (CAD) systems, which can provide additional support to radiologists and other medical professionals in making accurate and timely diagnosis.

1.6 PURPOSE

Tuberculosis is caused by the bacterium Mycobacterium tuberculosis. Although it mainly affects the lungs, it can spread to other body parts. When an infected individual coughs or sneezes, releasing droplets containing the bacterium, TB is disseminated through the air. The germs can infect and flourish in the lungs of another person who inhales these droplets. Once inside the body, the immune system can go after the bacteria. When the immune system successfully fights off an infection, the patient may not experience any symptoms. The bacteria can, however, sometimes defeat the immune system and result in active TB illness.

Effective TB treatment and transmission prevention depend on early and precise TB identification. Chest X-rays and other imaging tests are frequently used to identify TB because they can spot lung tissue anomalies that could be signs of the disease. However, due to the presence of bones, which can obscure the lung tissue and make it challenging to detect subtle changes, accurately diagnosing TB on X-rays can be challenging. Technology for bone suppression may be useful in this situation.

Through this project, we seek to determine the advantages of bone suppression in chest X-rays for the

automated diagnosis of lung diseases. As tuberculosis is one of the most common lung diseases, we have used its dataset to train our classification model to check the performance metrics of bone suppressed images versus normal images

Using computer algorithms,, bone suppression in X-rays eliminates or suppresses the appearance of bones from the image, revealing only soft tissue. This can increase the lung tissue's visibility,

making it simpler to spot any anomalies that might point to the presence of TB. We can increase the accuracy and speed of TB diagnosis by utilising bone suppression technology in combination with other detection techniques, such as machine learning algorithms, enabling earlier and more efficient treatment. This may lessen the disease's spread and enhance patient outcomes.

Thus as an end result of our project, we aim to highlight the merits of increasing the visibility of lung tissue and minimising the impact of bones on the picture. Bone suppression technology can significantly improve the accuracy of TB diagnosis on X-rays. Our capacity to identify and treat TB can be improved by utilising cutting-edge imaging technologies and machine learning algorithms, leading to better patient outcomes and a decrease in the spread of the illness.

CHAPTER 2

LITERATURE REVIEW

The base paper and a few others were used as reference for module implementation and constraint resolution. Demerits of the same have been mentioned in table format. Advancements in robust machine learning algorithms, underlying neural network architecture and the conceptualization of the designs and tools integrated in our study are all covered in this section.

2.1. ResNet

ReS-Net [Fig 2.1] is a neural net that was proposed for the task of image super-resolution, which aims to generate high-resolution images from inputs containing low-resolution images. ReS-Net uses residual connections and dilated convolutions to efficiently upscale images while preserving importantfeatures. The architecture includes a novel residual block called the Residual Encoder-Decoder (RED) block, which consists of multiple residual connections and a decoder path that gradually up samples the image. The residual block is defined as:

$$F(x) = x + H(x, \{Wi\}) \quad \text{.....(2.1)}$$

where H(x, {Wi}) is a residual function consisting of a series of convolutional layers and non-linear activation functions. The {Wi} denotes the learnable parameters (weights and biases) of the convolutional layers.

The addition of the input x to the output of the residual function H(x, {Wi}) creates a "shortcut connection" that allows gradients to flow directly through the network, mitigating the problem of vanishing gradients.

[1] Rajaraman et. al. compares models based on different algorithms, where the model based on ResNet -BS shows best performance. This model is then used for the classification of Tuberculosis.

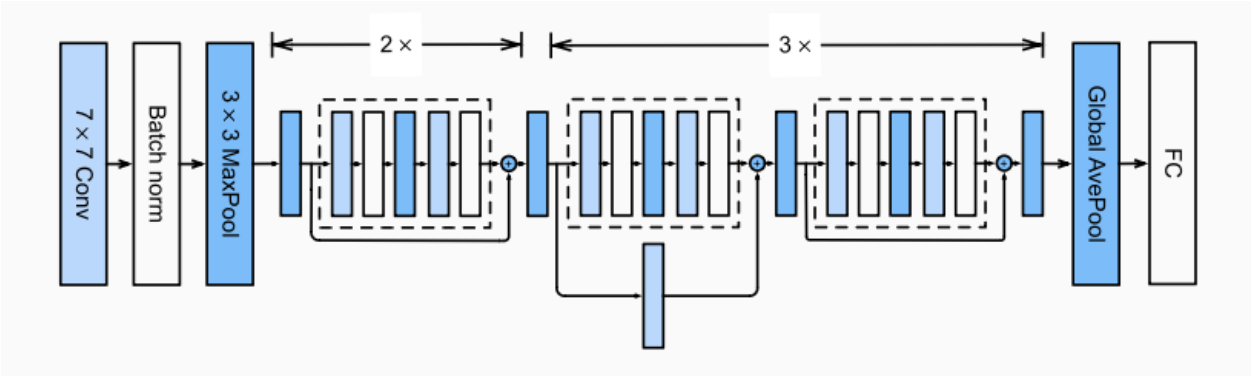


Fig 2.1: Architecture of Resnet

2.2. MTANN

To create images of bones, a machine learning model called Multi-Task Artificial Neural Network (MTANN) [Fig 2.2] was used by [3] Suzuki et al in 2006. [5] This model is capable of directly operating on pixel data and instead of using a sigmoid function, like the other models, uses a linear- output multilayer regression model as the activation function in the output layer. The use of a linear function has been shown to significantly improve the characteristics of the ANN when applied to continuous mapping of values in image processing.

The MTANN employs a massive-sub-region training scheme, which involves training with a large number of sub-region-pixel pairs. In the case of bone suppression in CXRs, the images are divided pixel by pixel into numerous overlapping sub-regions or image patches. The MTANN is trained usinga large number of input sub-regions and their corresponding teaching single pixels. The inputs to the MTANN are pixel values in a sub-region (or image patch) R that is extracted from an input image.

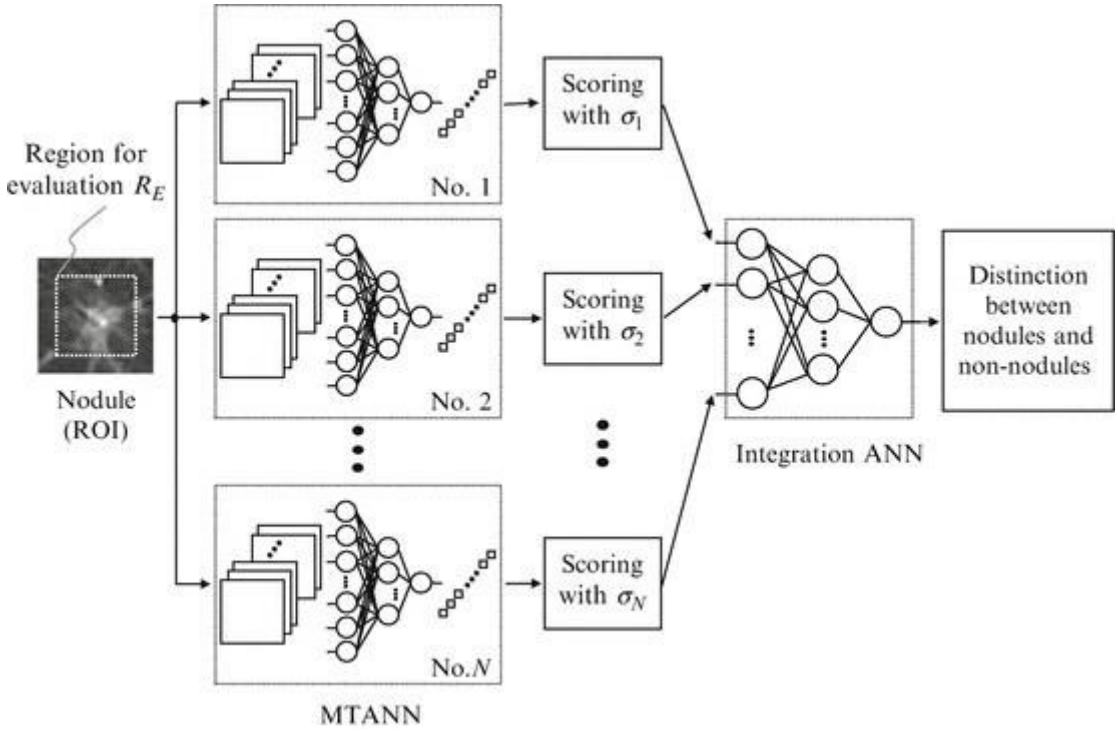


Fig 2.2: Architecture of MTANN

The MTANN was not effective in completely suppressing rib edges, ribs near the lung wall, and clavicles because bones in different locations have varying orientation, width, contrast, and density, and the capability of a single MTANN is limited. To enhance bone suppression in various locations,we developed an anatomically specific multiple-MTANN scheme. This scheme consists of eight MTANNs arranged in parallel to extend the capability of a single MTANN.

To suppress the ribs in a CXR taken in the ICU, the original image g(x,y) was subtracted by the bone image fb(x,y) generated by the anatomically specific multiple MTANN by g(x,y) with the masking image m(x,y) .

[4] Chen et. al. a bone suppression model was created and evaluated through image patches for tuberculosis diagnosis. The authors propose a rib suppression method based on MTANN that can be trained using massive sub-region-pixel pairs. The method involves dividing CXRs into overlapping

sub-regions and extracting single pixels corresponding to the input sub-regions from the teaching images. The MTANN is then trained using each input sub-region and its corresponding teaching single pixel.

The authors evaluated the effectiveness of the proposed method by comparing it to other methods on a publicly available dataset of CXRs. They found that their method improved the classification of textural abnormalities in CXRs and outperformed other methods in terms of accuracy. However some ridges in the lower and upper parts of the lungs were not suppressed each time.

2.3 Convolutional Filter

CF stands for Convolutional Filter [Fig 2.3]. It is a type of neural network architecture that is designed to learn spatial filters that are used to extract features from images. A CF is composed of multiple convolutional layers that learn to detect different visual patterns in an input image. The filters in each layer are learned through a process of backpropagation, where the network is trained on a large set of labeled images. The output of a CF is a feature map that represents the activation of each filter in the network for a given input image.

The bone-suppressed image can be obtained by performing element-wise multiplication between the input image I and the output of the bone suppression filter f(I), followed by normalization. Mathematically, it can be expressed as:

$$B = (I \otimes f(I)) / \max(I) \quad \text{.....(2.2)}$$

where $\otimes$ represents element-wise multiplication, and max(I) is the maximum pixel value in the input image, used for normalization.

The mathematical formula for the output of a CF can be expressed as:

$$Z(i,j,k) = b(k) + \sum\sum\sum X(m,n,l) * W(i-m+1, j-n+1, l, k) \quad \text{.....(2.3)}$$

where Z(i,j,k) is the output feature map at position (i,j) and channel k, b(k) is the bias term for channel k, X(m,n,l) is the input feature map at position (m,n) and channel l, W is the weight tensor of the CF with shape (h, w, c, f), h and w are the height and width of the kernel, c is the number of input channels, and f is the number of output channels.

[2] Matsubara et. al. have created CNF for spatial filtering with CNN .The authors first trained a CNF to learn the relationship between bone and soft tissue patterns in chest X-ray images. Then the filtered image is used to subtract from the original image and generate the bone suppressed image.

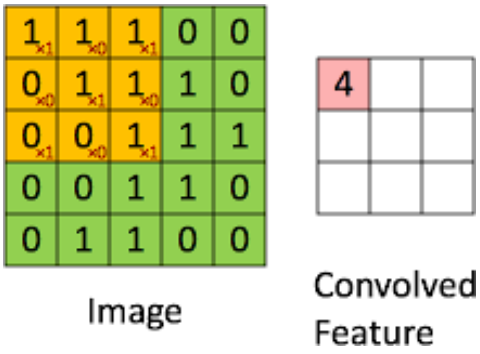


Fig 2.3: Convolutional Filter

2.4. GAN

Generative Adversarial Network (GAN) [Fig 2.4] is a type of neural net used in machine learning and artificial intelligence to generate new data corresponding to the training data. GANs consist of a generator network and a discriminator network. One network, called the generator, creates fake data samples that resemble real ones, while the discriminator tries to distinguish between the fake (generated) and the real samples. The two networks are trained together in a process known as

adversarial training, where the generator tries to produce better fake samples that can fool the discriminator, while the discriminator tries to become better at distinguishing real from fake samples.

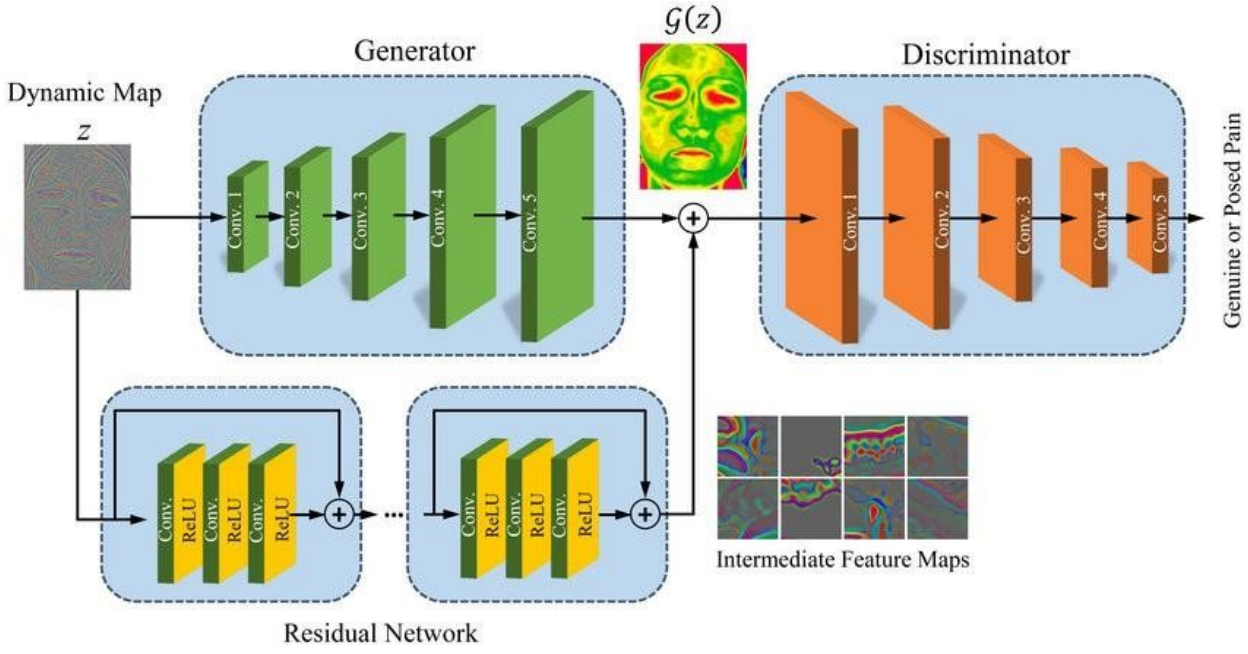


Fig 2.4: Architecture of a simple GAN

[6] Rani et. al. have proposed a GAN model optimized with loss functions for image retention and image quality. Model can be implemented for denoising of images.

The authors modified the discriminator and generator architectures of the pix2pix model to achieve a specific task. They combination of the discriminator and Wasserstein GAN along with Gradient Penalty resulted in improved performance and training stability. They used a combination of different loss functions, as generator.

2.5. Deep Feature Disentanglement Learning

[19] Deep feature disentanglement learning [Fig 2.5] is a technique in machine learning and computer

vision that aims to learn a disentangled representation of input data. The idea is to separate the underlying factors of variation in the data into distinct and independent features, which can then be manipulated or removed to modify the output of the system. In the context of bone suppression in chest radiographs, deep feature disentanglement learning is used to separate the bone structure from the soft tissue structure in the image, enabling the creation of a bone suppression image that preserves the important soft tissue features while removing the bone structure.

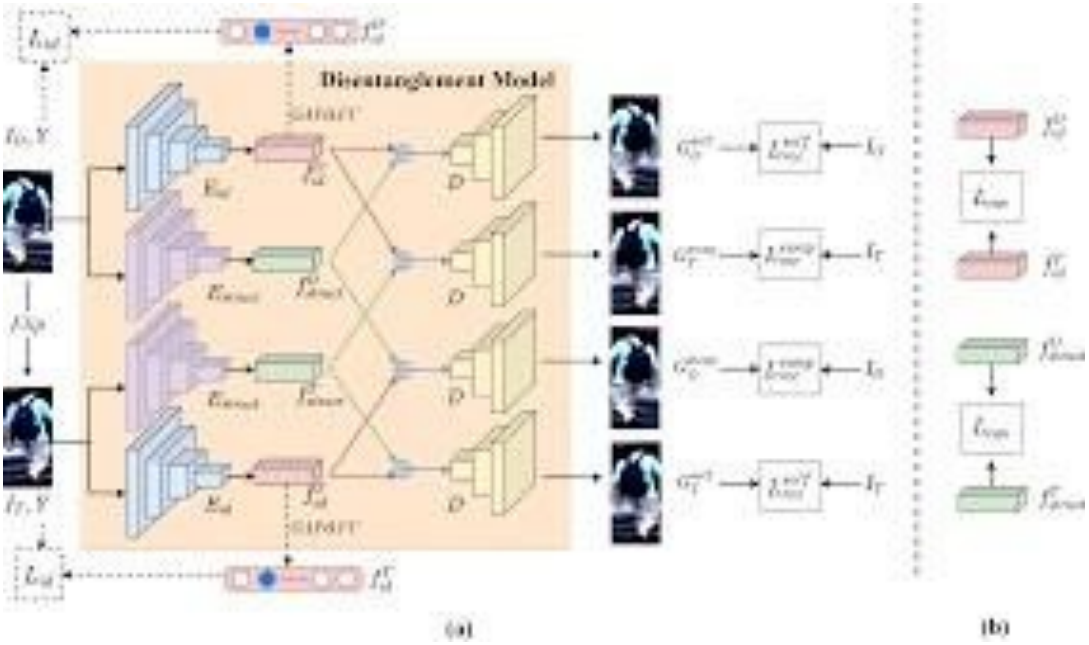


Fig 2.5: Deep Feature Disentanglement Learning

2.6. DES vs X-Rays

[7] Li et. al. compares the effectiveness of bone suppression imaging and dual-energy subtraction chest radiography in the detection of small lung cancers. The authors used two different methods of bone suppression imaging, a linear model-based approach, and a nonlinear model-based approach. The study evaluated the performance of each method on a dataset of chest X-rays with known lung cancer

cases. The authors found that both bone suppression imaging methods improved the detection of small lung cancers compared to dual-energy subtraction chest radiography. They also found that the nonlinear model-based approach provided the best performance overall. BS images together with normal x-rays increase detection of small lung cancers. DES approach provides a higher radiation dose and need special equipment. [Fig 2.6]

[8] Zarshenas A et. al. AS NNC, OFS NNC models were combined to create a virtual dual – energy image system. The authors provide a detailed overview of the DES-DR process and explain how it can improve the quality of chest radiographs by reducing the effect of overlying structures such as bones, mediastinal structures, and the diaphragm. The paper describes the imaging system and the steps involved in acquiring and processing dual-energy images, as well as the challenges that may arise during the procedure. The authors also discuss the potential clinical applications of DES-DR, such as the detection of lung nodules and the evaluation of pulmonary embolism. The paper concludes with a discussion of the advantages and limitations of DES-DR and its potential future developments.

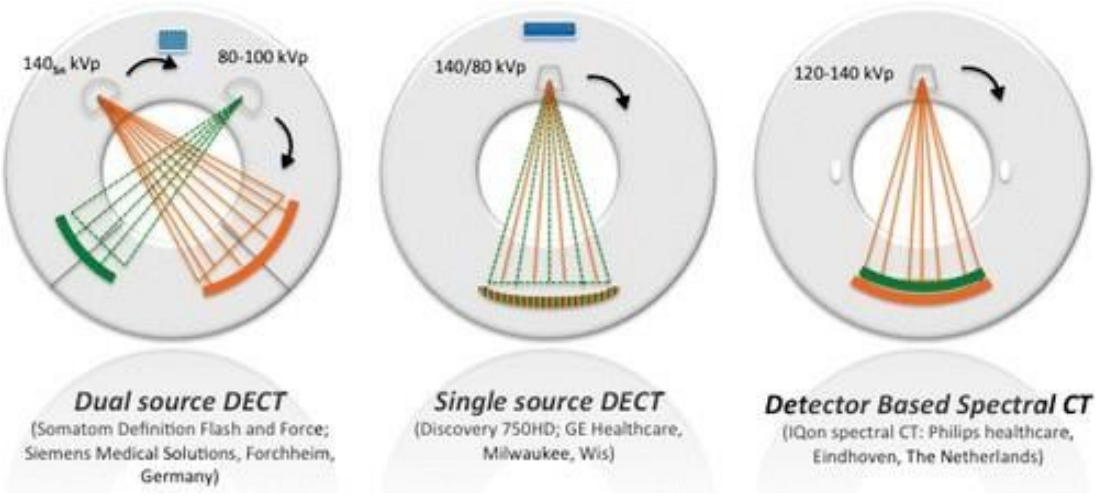


Fig 2.6: Visual representation of difference between Dual and single source CT

[9] This study highlights the disadvantages of using the dual-energy method in terms of its high radiation doses, lack of lateral view images and it cannot be used in portable chest radiography.

[10] Maduskar et. al. tested their Bone suppression image model for CAD detection with slight improvement in detection of abnormality as compared to the CAD system trained with regular chest x-rays.

2.7. CT Scans vs X-Rays

[11] The study concluded that screening for cardiovascular disease with CT in the Danish population studied did not result in a net benefit and was associated with more costs than benefits. Therefore, the authors suggested that routine screening with CT [Fig 2.7] should not be recommended in this population. CT Scans were shown to be an expensive screening method.

[12] Power et al. discusses the risks associated with the use of computed tomography (CT) scans and the factors that contribute to patient risk perception. The authors explain that while CT scans provide valuable diagnostic information, they also expose patients to ionizing radiation, which can increase the risk of cancer. The paper reviews the current understanding of the risks associated with CT scans and highlights the uncertainties in risk estimation.

[13] The authors review the current guidelines for CT imaging, including the use of low-dose protocols and the appropriate use of CT scans. They also discuss the potential for new technologies to reduce radiation exposure, such as iterative reconstruction algorithms and photon-counting detectors.

Overall, the authors conclude that while the use of CT imaging has revolutionized medical diagnosis and treatment, it is important to carefully consider the potential harm of radiation exposure to patients.

They emphasize the importance of minimizing radiation exposure through appropriate use of CT imaging and the use of new technologies to reduce radiation dose.

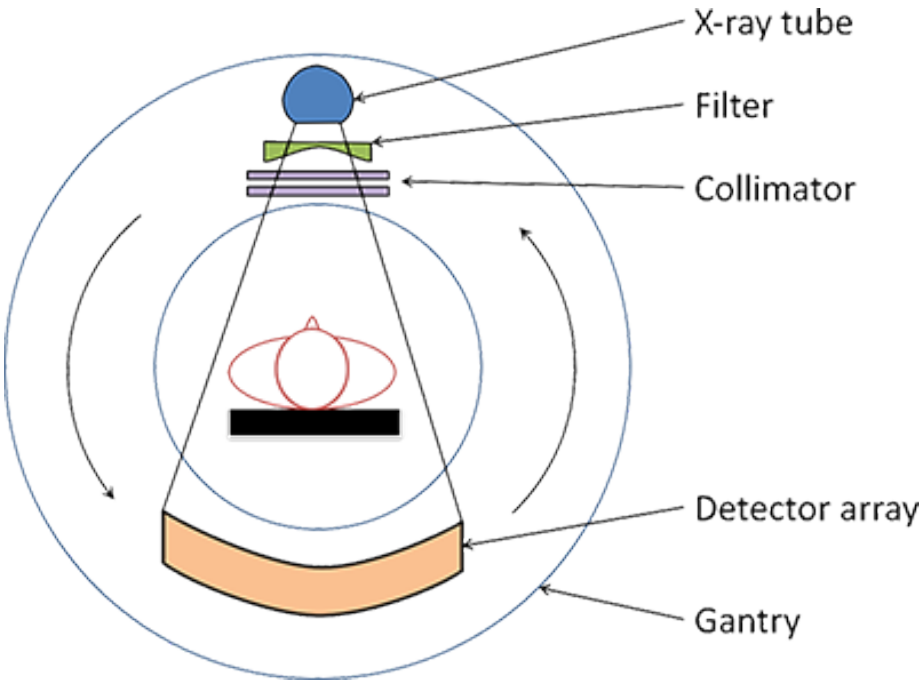


Fig 2.7: CT mechanism

Table 2.1 : Limitations of Related Work

Author	Title	Journal Published	Year	Limitations
Rajaraman et. al.	"Chest X-ray Bone Suppression for Improving Classification of Tuberculosis-Consistent Findings"	*Diagnostics* (Basel).	2021	Limited data with low diversity was used for training resulting in low generalization for real-world images

Gusarev et. al.	“ Deep learning models for bone suppression in chest radiographs”	2017 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)	2017	Lack of in-depth research with optimization friendly properties
Chen S. et. al.	“Enhancement of chest radiographs obtained in the intensive care unit through bone suppression and consistent processing.”	Phys Med Biol.	2016	Less Optimized parameters, failure of bone suppression due to bone suppression in only lung segmented areas.
Zarshenas A et. al.	" Separation of bones from soft tissue in chest radiographs: Anatomy-specific orientation-frequency-specific deep neural network convolution”	Med Phys.	2019	Hyper-parameters can be tuned for better performance. Multiple layers used however a single layer can be used to achieve better results.
Whitman GJ et. al.	"Dual-energy digital subtraction chest radiography: technical considerations”	Curr Probl Diagn Radiol	2002	DUI uses special added equipment and exposes patient to high radiation.

Hogeweg et. al.	"Rib suppression in chest radiographs to improve classification of textural abnormalities"	Proceedings Volume 7624, Medical Imaging 2010: Computer-Aided Diagnosis	2010	Ridges are not completely suppressed.
Li et. al.	"Small Lung Cancers: Improved Detection by Use of Bone Suppression Imaging—Comparison with Dual-Energy Subtraction Chest Radiography"	Radiology	2011	Use of selected cases under controlled environment.

CHAPTER 3

PROPOSED METHODOLOGY

In this chapter, we have provided elaborate details of the datasets used for the training of our models and the architecture and working behind the autoencoder model used for bone suppression and the classification models used for Tuberculosis Diagnosis. The entire project is divided into 3 modules and we will discuss them in the order of execution. The first module consists of obtaining and preparing the dataset for autoencoder. The second module is the design and architecture of the autoencoder model. The third module is the development of the classification models for tuberculosis diagnosis.

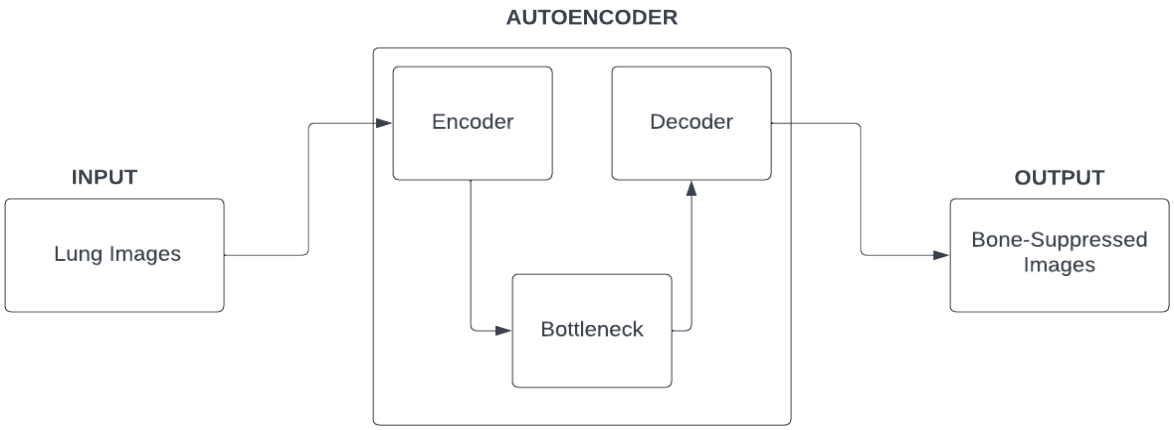


Fig 3.1: Block Diagram

3.1. MODULE 1 : DATASET

For the training of our bone suppression model, we have combined two medically credible datasets each containing labeled images of Chest X-Rays and their bone-suppressed images. The two datasets contain Chest-X Rays with contrasting background images, both proving a wide range of pixel colors

for the training of the model, allowing it to adapt to any kind of input image for conversion.

The first dataset [1] [Fig 3.2] used consists of CXRs with light background and it consists of a total of 4080 augmented images.

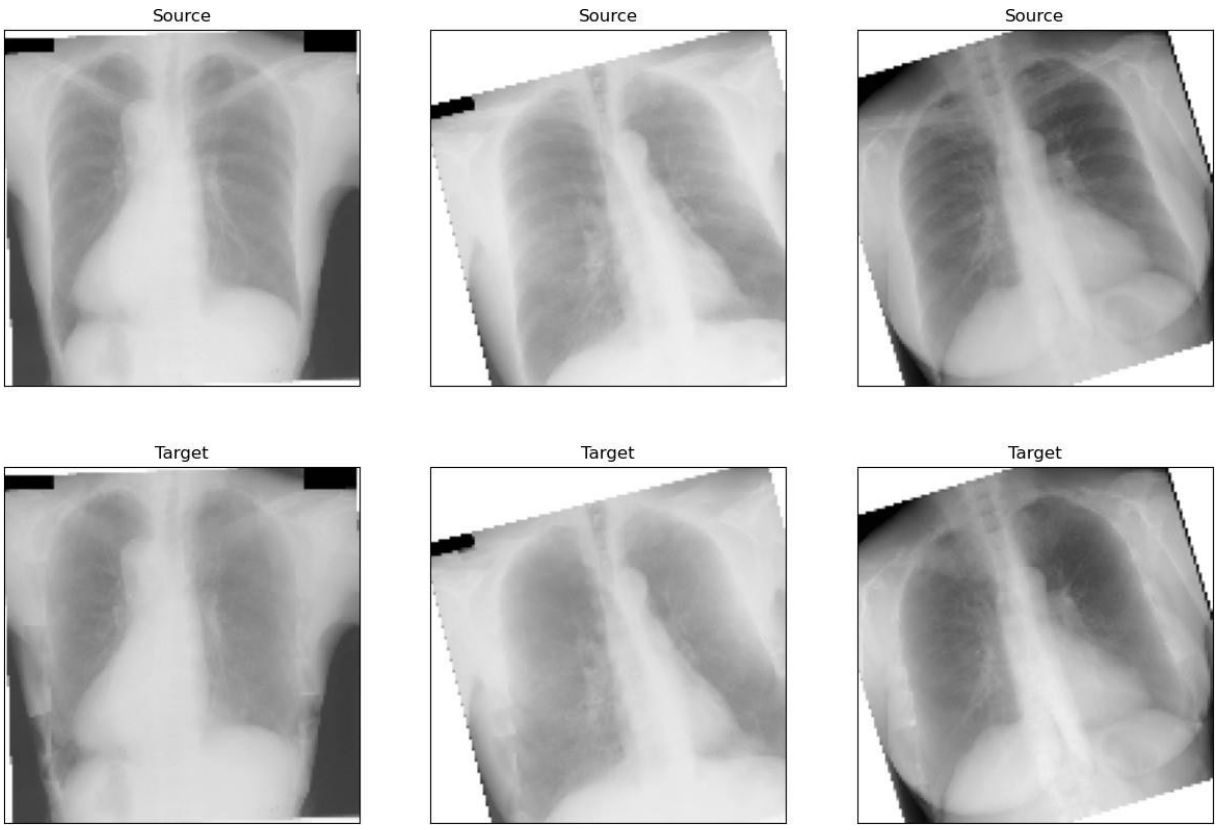


Fig 3.2: Training images for first dataset

The second dataset [16] [Fig 3.3]used consists of CXRs with dark background

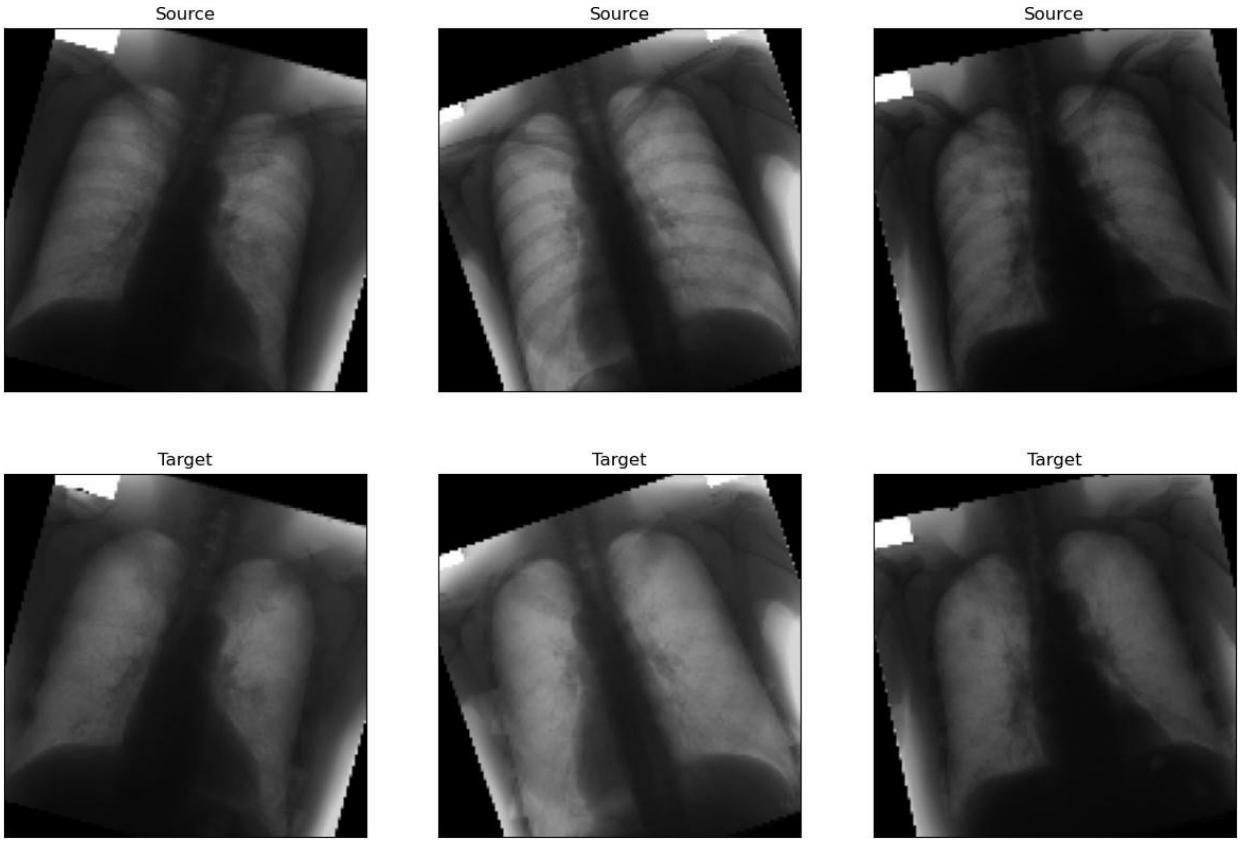


Fig 3.3: Training images from second dataset

The combined dataset consists of both types of images with index shuffled, allowing the model to adapt better to the images given [Fig 3.4]

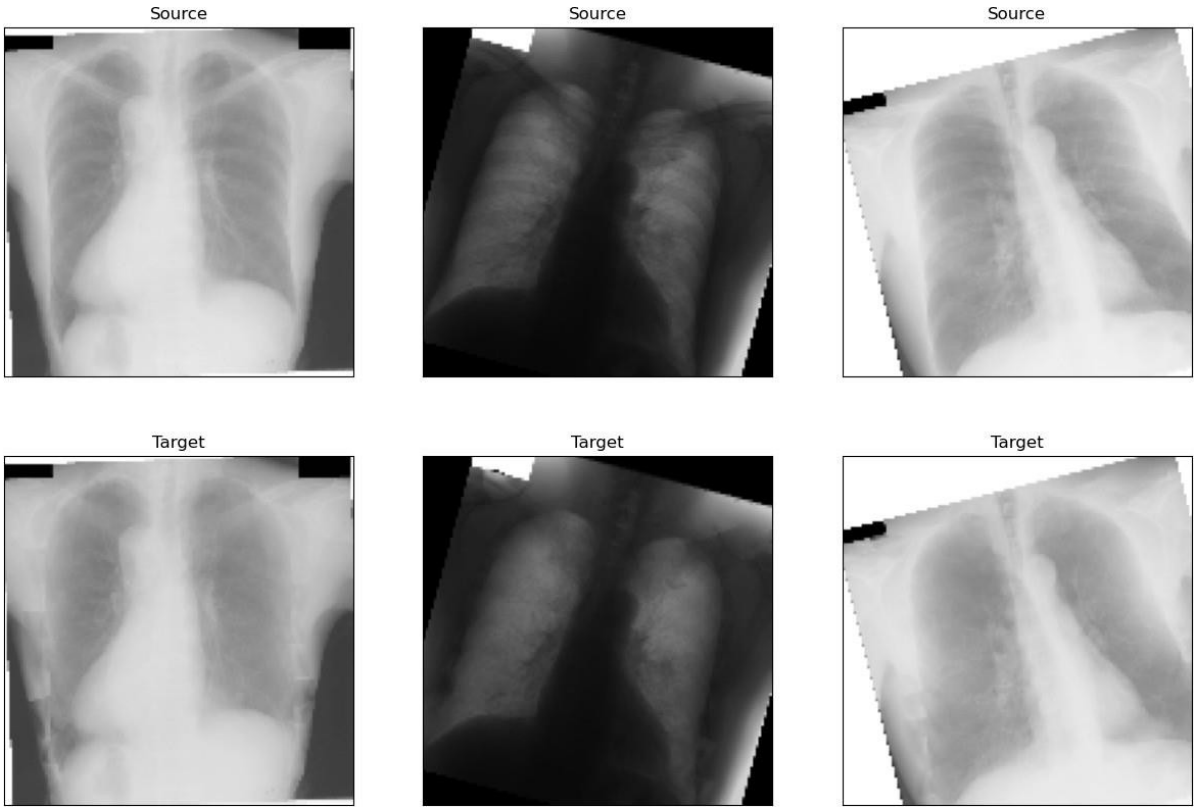


Fig 3.4: Training images from combined dataset

3.2 MODULE 2 : AUTOENCODER

Autoencoders are a type of neural network that have gained popularity in computer vision applications due to their ability to learn compact and useful representations of input data. They are particularly useful in image processing tasks where feature extraction and compression are necessary, such as image denoising, compression, and reconstruction.An encoder and a decoder are the two primary partsof an autoencoder's architecture. The encoder converts an input image to the latent space, a representation with less dimensions. This latent representation is then used by the decoder to produce areconstructed image that is comparable to the input image. The model learns a meaningful andcondensed representation of the input data by attempting to minimize the difference between the

original image and the reconstructed image during training

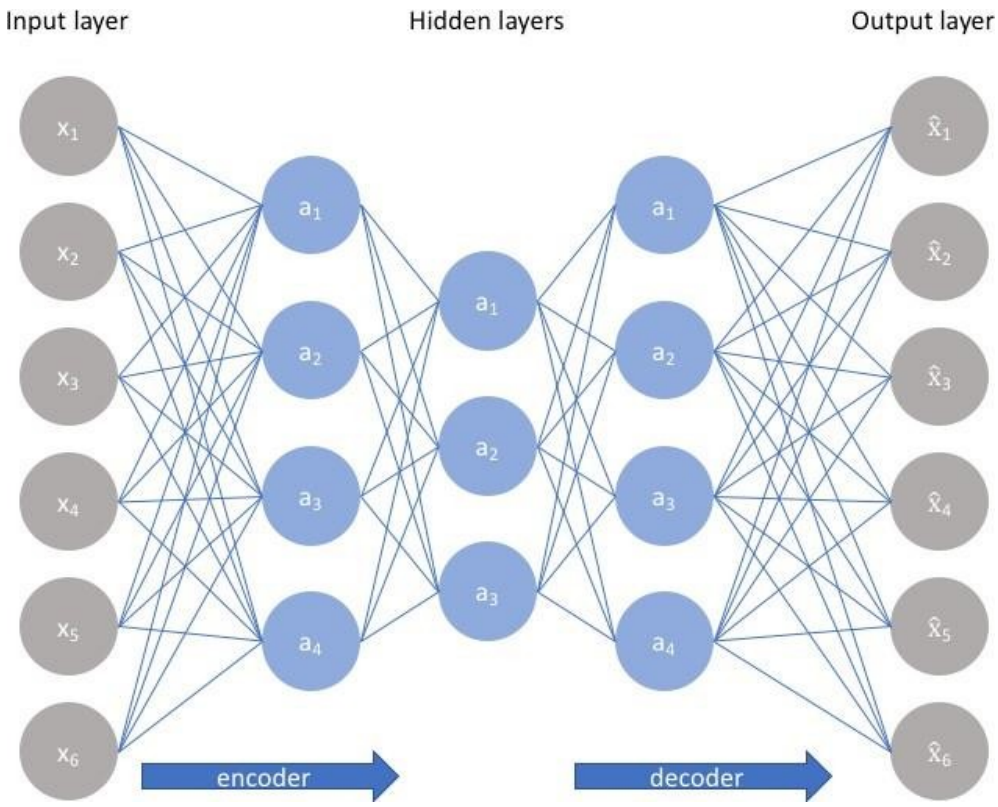


Fig 3.5: Architecture of Autoencoder

One advantage of autoencoders in computer vision is their ability to learn data-driven feature representations. They can automatically acquire relevant features from the input data, as opposed to hand-crafted features, which need domain-specific knowledge and can be time-consuming to build. Due to this, they are especially beneficial for tasks where domain-specific knowledge is scarce or challenging to come by.

The ability of autoencoders to deal with lost or damaged data is another benefit. The model may frequently produce acceptable reconstructions even when portions of the input image are missing or corrupted since it is taught to rebuild the input image from the latent representation

Image compression, denoising, segmentation, and object recognition are just a few of the many computer vision applications that autoencoders can be used for. Autoencoders can be used in image compression to shrink an image while keeping crucial details, which is beneficial in applications where storage or transmission bandwidth is constrained. In order to create clean copies of noisy images, autoencoders can be trained on those images. Autoencoders can provide pixel-level masks for various objects in an image during segmentation. Autoencoders can also be employed in objectrecognition to teach a compact and discriminative representation of the input images that can be applied to classification problems.

In this project, to create a bone suppression autoencoder model, the input image is an X-ray image that contains bones and other structures, and the goal of the model is to remove the bones from the image while preserving the other structures. This is achieved by training the model on a dataset of paired X-ray images, one with bones and one without. During training, the model learns to encode the bones in the input image and then suppress them in the output image using the two decoders. The final output is an image that retains the important structures while removing the bones, which can be useful for a variety of medical imaging applications.

Three often used metrics in autoencoders for assessing the quality of the reconstructed image are reconstruction loss, Mean Absolute Error (MAE), and Mean Squared Error (MSE). All three measures compare the differences between the input and output images, but they each have unique traits and are best suited for certain kinds of issues. While MAE and MSE are specific metrics that assess the difference between the pixel values of the input and output images, reconstruction loss is a more general term that refers to any metric used to assess the quality of the reconstructed image. When the absolute difference between pixel values matters, the MAE is useful, whereas the MSE is better suited for issues where the squared difference matters more.

3.2.1 Mean Absolute Error

As a performance metric for our Autoencoder Model, we have used MAE. It is a commonly used metric to evaluate the performance of regression models, including autoencoder neural networks. In the context of autoencoders, MAE measures the average absolute difference between the original inputdata and the reconstructed output data, across all input examples in the test dataset.

The formula for calculating MAE is:

$$MAE = (1/n) * \Sigma|yi - \hat{y}i| \quad [3.1].$$

where yi is the actual value, ŷi is the predicted value, and n is the number of data points.

MAE is a useful metric in autoencoder neural networks because it is robust to outliers and can provide a better estimate of the true error compared to mean squared error (MSE) in some cases. However, MAE does not take into account the relative importance of errors, so it may not always be the most appropriate metric depending on the specific application. In some cases, MSE or other metrics may be more appropriate.

3.2.2 Mean Squared Error

We have used MSE as another performance metric to identify the shortcomings and finetune our autoencoder model accordingly. Mean squared error is a commonly used measure of the average of the squared differences between the predicted and actual values in statistical models. The higher the MSE, the greater the difference between the predicted and actual values, and thus, the higher the level of error in the model. when the model's predictions are exactly equal to the actual values, the MSE is zero. As the model's error increases, the MSE value also increases. Additionally, the mean squared

error is also referred to as the mean squared deviation (MSD) because it is a measure of the average deviation or error between the predicted and actual values.

3.2.3 Reconstruction Loss The reconstruction loss is a measure of how well the autoencoder is able to reconstruct the original input data from the encoded representation. It is typically computed as the difference between the original input data and the reconstructed output data, using a suitable distance metric such as mean squared error or binary cross-entropy. The goal of the autoencoder is to minimize the reconstruction loss during training, which encourages the encoder to learn a compressed representation of the input data that preserves the most important features of the data. Once the autoencoder is trained, the encoder network can be used as a feature extractor for other tasks, such as classification or clustering.

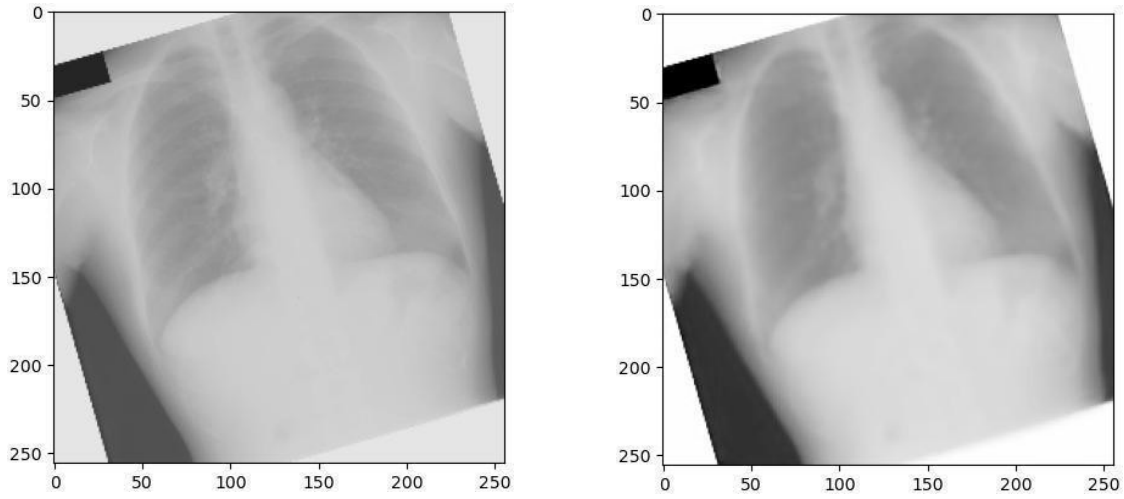


Fig 3.6: (left) Input , (right) Output

There are several types of autoencoders used in computer vision, including:

1. **Standard autoencoders** : This is the most basic form of an autoencoder consisting of an encoder, a bottleneck and a decoder. An input image is compressed by the encoder into a lower-dimensional

representation, which is then used by the decoder to reconstitute the original image. The mean squared error between the input and output images often serves as the loss function.

2. **Convolutional Autoencoder** : Convolutional layers [Fig 3.7] are used in the encoder and decoder of this sort of autoencoder instead of fully linked layers. Compared to fully connected layers, convolutional layers are better able to handle spatial information in images and can learn more complicated characteristics.

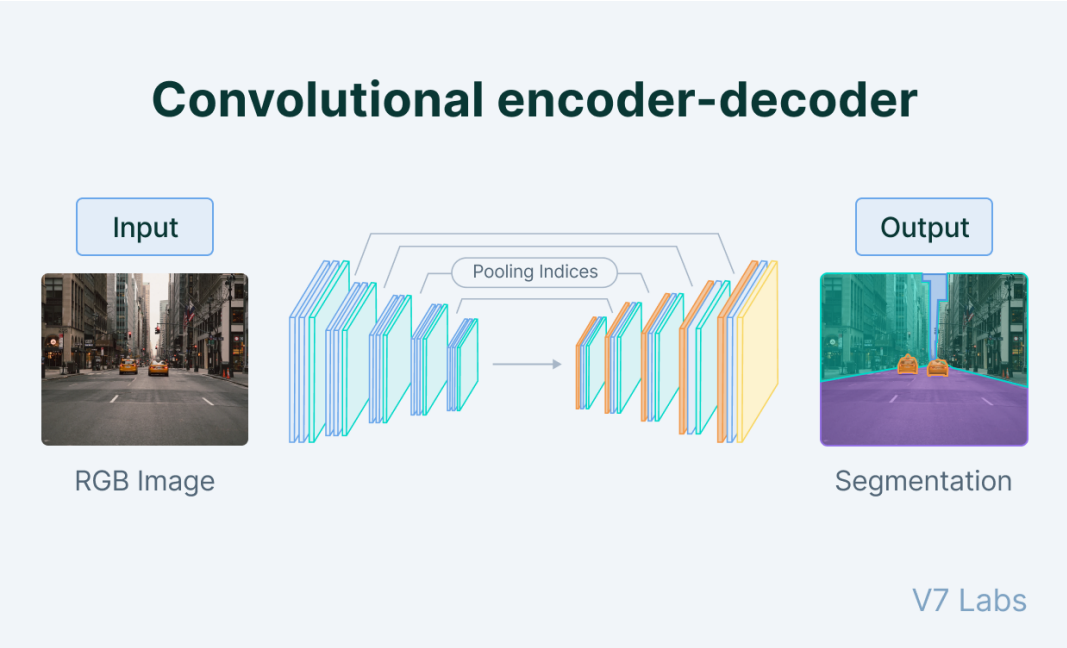


Fig 3.7: Architecture of Convolutional Autoencoder

3. **Denoising Autoencoder** : With the help of learning a representation that captures the underlying structure of the input data, this kind of autoencoder can be used to reduce noise in photos. The network receives a noisy image as input, and outputs a denoised image [Fig 3.8]. Reconstruction loss and a regularisation term that penalises departures from the learned structure are typically combined to form the loss function.

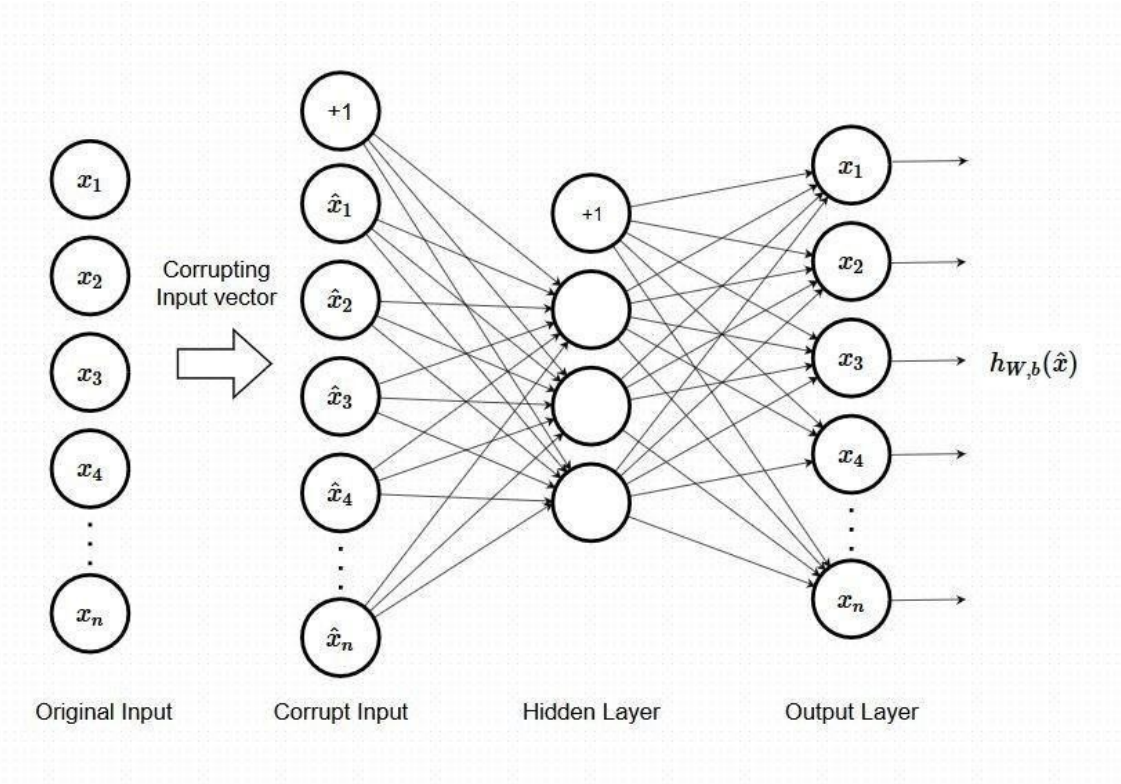


Fig 3.8: Architecture of Denoising Autoencoder

4. **Variational Autoencoder** : Instead of learning a deterministic mapping, this kind of autoencoder learns a probabilistic distribution of the input data. When creating new images, it is frequently used to sample from the learned distribution [Fig 3.9]. The loss function encourages the learning distribution to match a predetermined distribution by combining a regularisation term with the reconstruction loss.

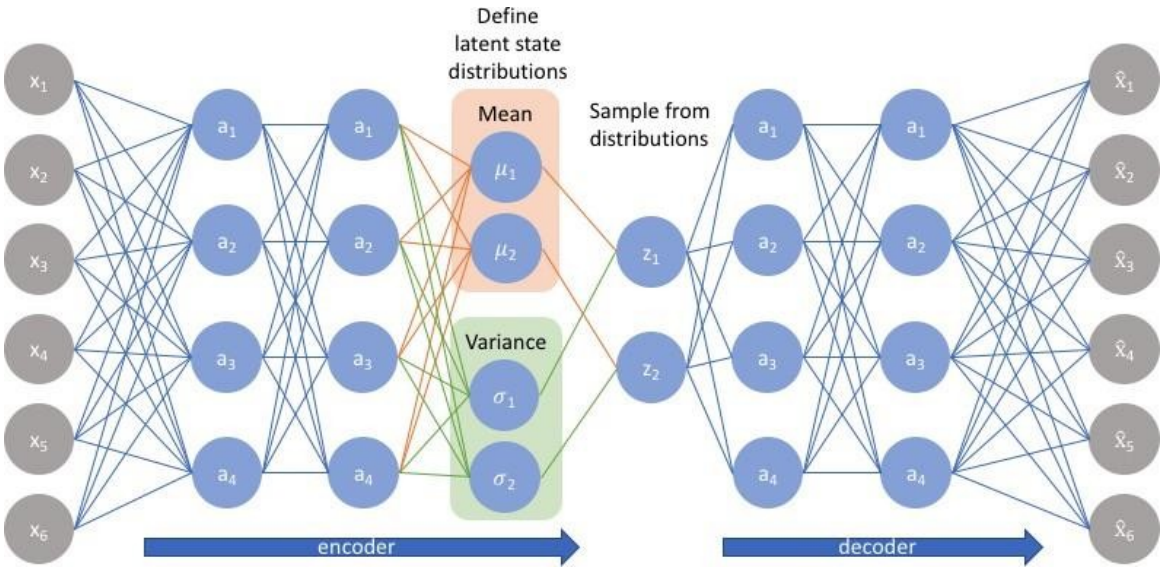


Fig 3.9: Architecture of Variational Autoencoder

5. **Adversarial Autoencoder** : The principles of adversarial networks and autoencoders are combined in this kind of autoencoder [Fig 3.10]. It gains the ability to convert the input data into a lower-dimensional representation that retains the data's details while also being similar to samples taken from a known distribution. The loss function encourages the learned representation to match the previous distribution by combining an adversarial loss with the reconstruction loss.

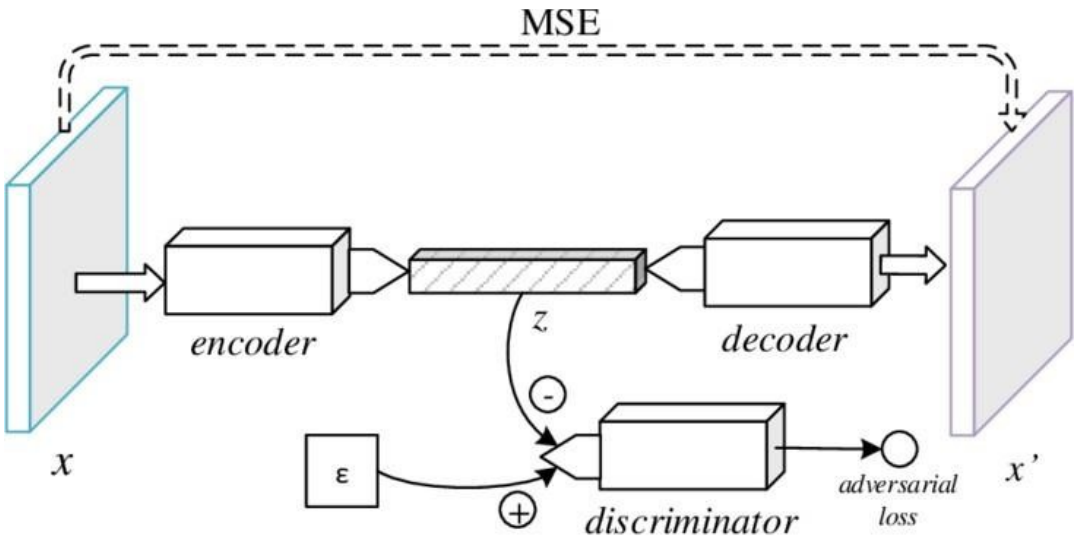


Fig 3.10: Architecture of Adversarial Autoencoder

3.3. MODULE 3 : CLASSIFICATION

For the classification of Tuberculosis in Chest X-Ray images, we have built our own binary classifier which classifies the image into normal (0) or Tuberculosis (1). We have used the basic principles of Convolutional Neural Networks for designing and building this binary classifier, and have used the same model to train using both the datasets for easy comparison of performance.

3.3.1 Convolutional Neural Networks:

Typically, convolutional layers are followed by one or more dense layers in CNNs for binary classification. They are in charge of extracting features from the input image, while dense layers are used to decide on the classification at the end of the process using the features that have been extracted.

The generated feature maps are then flattened into a one-dimensional vector using the flatten layer following a number of convolutional layers. This process is required to convert the output of the convolutional layers into a format that the dense layers can understand. In order to do classification based on the collected characteristics, the flattened vector is then passed through one or more dense layers, which are completely linked layers.

To avoid overfitting, CNNs frequently employ dropout layers. Overfitting happens when a model gets overly complicated and cannot generalize, it starts to fit too closely to the training data instead of the underlying patterns. During training, certain neurons in a layer are randomly removed using the regularisation approach known as "dropout," which forces the network to learn more robust characteristics that are less susceptible to minute changes in the input.

The final dense layer in a CNN for binary classification typically uses a sigmoid activation function. The reason the sigmoid function is frequently employed in this layer is that it yields an output in the range of 0 to 1, which may be interpreted as the likelihood that the input image belongs to either of the two classes. Setting a threshold value will classify any output above the threshold as belonging to one class, and any output below the threshold as belonging to the other. This output will then be used to make a binary classification decision.

Any input value is translated by the "S"-shaped curve [Fig 3.11] of the sigmoid function to a value between 0 and 1. The ability to forecast the likelihood of a binary outcome makes it a strong option for binary classification problems. The sigmoid function's output can also be thought of as a measure of how confident the model is in its prediction; larger values signify more confidence.

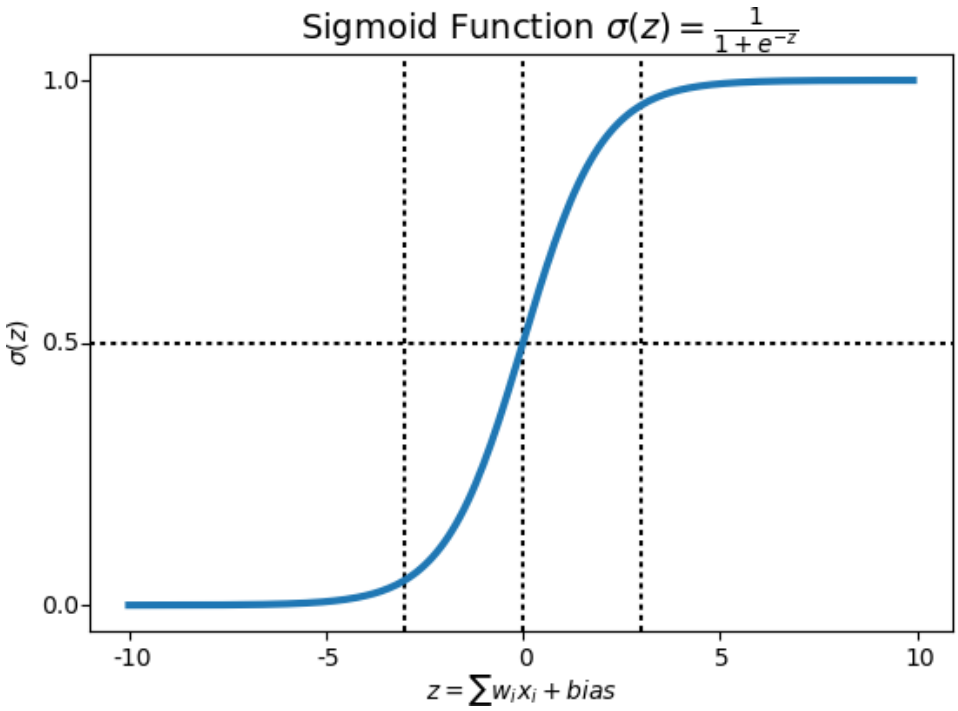


Fig 3.11: Graph of Sigmoid Function used in final dense layer

The sigmoid function has the additional benefit of being differentiable, which enables backpropagation to be used during training. The model weights are updated during training using a method called backpropagation, which calculates the gradient of the loss function with respect to the model parameters. Calculating the gradients required for backpropagation is simple because of the sigmoid function's differentiability.

```
model = models.Sequential([
    data_augmentation,
    layers.Conv2D(28, (3, 3), activation='relu', input_shape=(28, 28, 3)) ,
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu')

])
```

```
model.add(layers.Flatten())
model.add(layers.Dense(640, activation='tanh'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(564, activation='tanh'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(64, activation='tanh'))
model.add(layers.Dense(64, activation='sigmoid'))
model.add(layers.Dense(2))
model.summary()
```

Fig 3.12: Code snippet of the binary classifier

3.3.2 Optimizer

The optimizer algorithm we have used for training our model is Adam (Adaptive Moment Estimation). Adam is a variation of the stochastic gradient descent (SGD) technique that is particularly suited for large-scale, complicated issues because it dynamically modifies the learning rate of each weight in the network during training. Adam is frequently used in binary classification jobs because it enables faster network convergence, which is crucial for real-time applications where quick decision-making is essential. By modifying the learning rate for each weight in the network based on its historical gradient

information, it also aids in lowering the danger of overfitting, a prevalent issue in deep learning.

Adam has a major benefit over other optimization algorithms like SGD and Adagrad because it can deal with noisy or sparse gradients, which are frequently encountered in deep learning applications. In order to lessen the effect of noisy or sparse gradients on the optimisation process, Adam estimates the first and second moments of the gradient using exponential moving averages. The gradient's varianceis represented by the second moment, whereas the gradient's mean is represented by the first moment.

3.3.3 Loss Function

The loss function we have used for training and keeping a track of the performance of our model is Sparse Categorical Entropy (SCE). SCE is a modification of the categorical cross-entropy loss function where the target is represented as an integer rather than a one-hot vector and is intended to handle sparse target classes. The output of the final dense layer of the CNN is a probability distributionover all conceivable classes, and the target class is represented as an integer in the loss function.

SCE is superior to other loss functions like binary cross-entropy because it makes it possible to train CNNs effectively on massive datasets with sparse target classes. For binary classification problems, where one class may be much less prevalent than the other, this is crucial. In these situations, binary cross-entropy can favour the more prevalent class, which would result in subpar performance on the minority class. SCE resolves this issue by computing the cross-entropy loss for each target class independently and considering each target class as a separate binary

classification task. As a result, even if the classes are imbalanced, the CNN may learn to distinguish between them because the loss function is not biased towards any one class.

The ability to utilize non-sigmoid activation functions in the last dense layer, such as the softmax

activation function, is another benefit of SCE. By establishing a threshold value, the softmax function generates a probability distribution over all feasible classes that can be used to decide on a binary classification.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 AUTOENCODER

The two often used performance metrics for assessing the quality of reconstructed pictures produced by an autoencoder are -MSE and MAE.

The MAE calculates the average absolute difference between the input and output images' pixel values. Since it gives equal weight to all differences, positive or negative, it is useful when the absolute difference between pixel values is significant. By averaging the absolute differences between the matching pixels in the input and output images, the MAE is determined. A lower MAE suggests a higher-quality reconstruction.

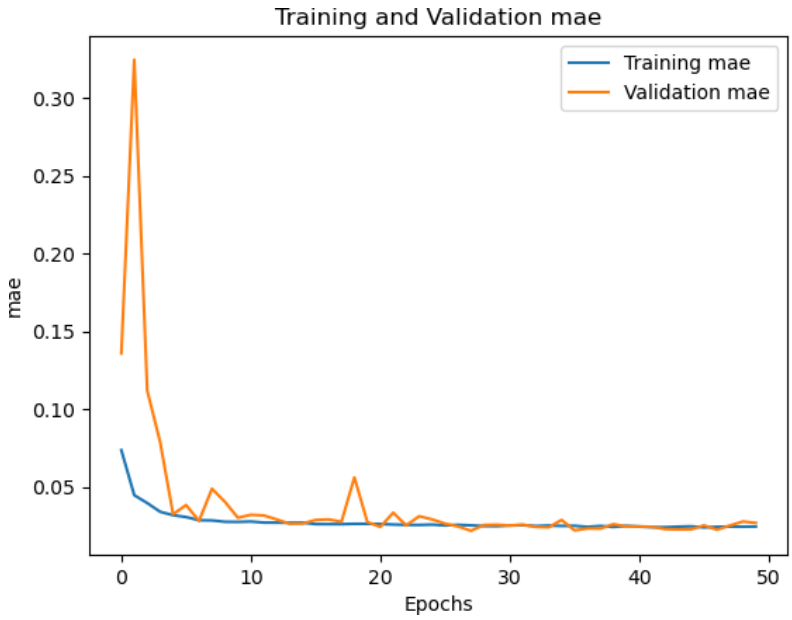


Fig 4.1: Graph of Training MAE vs Validation MAE

On the other hand, The average squared difference between the pixel values of the input and output images is measured by the MSE. Since it squares the variations in pixel values, it is more sensitive to large differences than the MAE. The average of the squared deviations between the matching pixels in the input and output images is used to calculate the MSE. Additionally, a lower MSE indicates higher reconstruction quality.

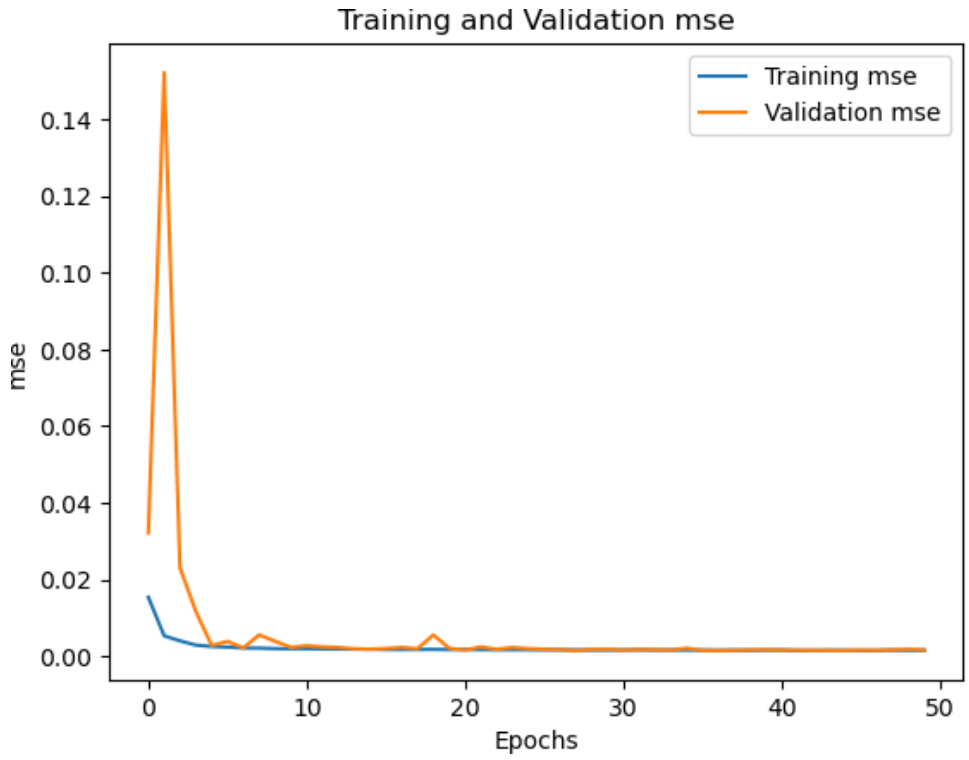


Fig 4.2: Graph of Training MSE vs Validation MSE

Checking loss

```
interval_epochs = [0, 49]
for e in interval_epochs:
    print("epoch = {}\tLoss = {:.5f}\tValidation_Loss = {:.5f}".format(e+1,autoencoder_train.history['loss'][e],autoencoder_train.history['val_loss'][e]))
```

```
epoch = 1       Loss = 0.02059  Validation_Loss = 0.03037
epoch = 50      Loss = 0.00092  Validation_Loss = 0.00124
```

Fig 4.3: Decrease of loss over epochs

4.2 CLASSIFICATION

Binary Classifier for Normal Chest X-Ray images

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential (Sequential)      (None, 28, 28, 3)         0
_________________________________________________________________
conv2d (Conv2D)              (None, 26, 26, 28)        784
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 28)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        16192
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 640)               369280
_________________________________________________________________
dropout (Dropout)            (None, 640)               0
_________________________________________________________________
dense_1 (Dense)              (None, 564)               361524
_________________________________________________________________
dropout_1 (Dropout)          (None, 564)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                36160
_________________________________________________________________
dense_3 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 130
=================================================================
Total params: 825,158
Trainable params: 825,158
Non-trainable params: 0
_________________________________________________________________
```

Fig 4.4: Model summary for first classification model

```
Epoch 1/10
191/191 [==============================] - 6s 24ms/step - loss: 0.5717 - accuracy: 0.6560 - val_loss: 0.2885 - val_accuracy: 0.9187
Epoch 2/10
191/191 [==============================] - 4s 23ms/step - loss: 0.2583 - accuracy: 0.8977 - val_loss: 0.2445 - val_accuracy: 0.8923
Epoch 3/10
191/191 [==============================] - 4s 22ms/step - loss: 0.2108 - accuracy: 0.9154 - val_loss: 0.1329 - val_accuracy: 0.9451
Epoch 4/10
191/191 [==============================] - 4s 23ms/step - loss: 0.1806 - accuracy: 0.9304 - val_loss: 0.1396 - val_accuracy: 0.9538
Epoch 5/10
191/191 [==============================] - 4s 23ms/step - loss: 0.1528 - accuracy: 0.9452 - val_loss: 0.0758 - val_accuracy: 0.9780
Epoch 6/10
191/191 [==============================] - 5s 27ms/step - loss: 0.1364 - accuracy: 0.9504 - val_loss: 0.0939 - val_accuracy: 0.9703
Epoch 7/10
191/191 [==============================] - 4s 23ms/step - loss: 0.1293 - accuracy: 0.9516 - val_loss: 0.1437 - val_accuracy: 0.9440
Epoch 8/10
191/191 [==============================] - 4s 22ms/step - loss: 0.1069 - accuracy: 0.9650 - val_loss: 0.0705 - val_accuracy: 0.9758
Epoch 9/10
191/191 [==============================] - 4s 23ms/step - loss: 0.1081 - accuracy: 0.9604 - val_loss: 0.0598 - val_accuracy: 0.9813
Epoch 10/10
191/191 [==============================] - 4s 22ms/step - loss: 0.1171 - accuracy: 0.9567 - val_loss: 0.0596 - val_accuracy: 0.9791
```

Fig 4.5: accuracy and loss across 10 epochs for first binary classifier

```
test_data = []
image='/Users/utkarsh/majorProject/DS/bone_suppressed/image120.png'
img = cv2.imread(str(image))
img = cv2.resize(img, (28,28))
if img.shape[2] ==1:
    img = np.dstack([img, img, img])
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img=np.array(img)
img = img/255
test_data.append(img)

# Convert the list into numpy arrays

test_data1 = np.array(test_data)
a=model.predict(np.array(test_data1))
if(np.argmax(a)):
    print('Tuberculosis')
else : print('Normal')
```

```
2023-03-18 11:51:17.608812: W tensorflow/tsl/platform/profile_utils/cpu_u
2023-03-18 11:51:17.693477: I tensorflow/core/grappler/optimizers/custom_
1/1 [==============================] - 0s 225ms/step
Normal
```

Fig 4.6: Inference code for first model

Binary Classifier for Bone Suppressed images

In the following graph, the accuracy vs val_accuracy graph shows the performance of a CNN during training and validation. The accuracy metric counts the percentage of accurate binary classification forecasts the model made based on the training set. The val_accuracy metric, on the other hand, gauges the proportion of accurate binary classification predictions the model made based on validationdata.

A visual depiction of the performance of the CNN during training and validation is provided by the accuracy versus val_accuracy graph. In a perfect world, the accuracy and val_accuracy metrics would both rise with time and reach a high value. If there is a significant difference between the accuracy and val_accuracy metrics, the model may be overfitting the training set and further regularisation methods, like dropout or weight decay, may be required to enhance generalisation.

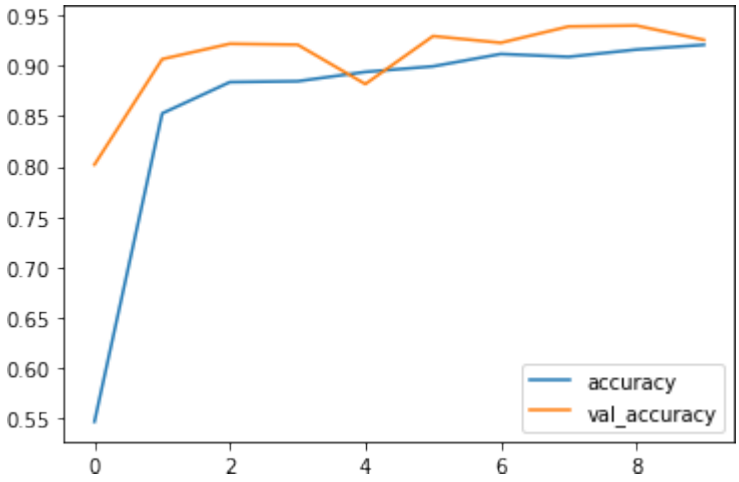


Fig 4.7: accuracy vs val_accuracy graph

In the next graph, the loss vs val_loss graph shows the performance of a convolutional neural network (CNN) during training and validation in terms of the loss function. The CNN is tuned during training to minimise the loss function, which raises the precision of the binary classification predictions on the training set. The val_loss metric calculates the CNN's loss on a different validation dataset and is used to assess how well the model generalises.

The loss versus val_loss graph gives a visual depiction of how the CNN performed in terms of the loss function during training and validation. The loss and val_loss metrics should, in theory, both drop with time and converge to a small value. However, if the model grows too complex, it may begin to overfit the training set, which would lead to a reduction in val_loss.

Loss and val_loss curves that decline and converge simultaneously show that the CNN is not overfitting and is successfully generalising to fresh data. If the loss curve keeps falling but the val_losscurve rises, the model may be overfitting the training set and failing to generalise to fresh data. In this situation, regularisation methods like dropout or weight decay might be required to avoid overfitting.

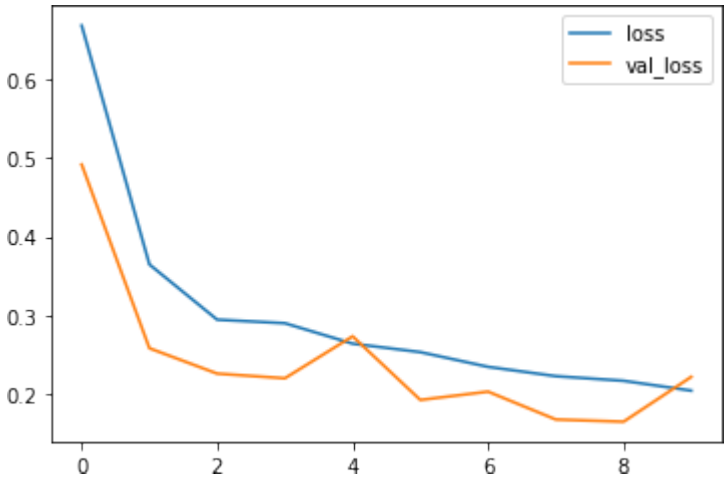
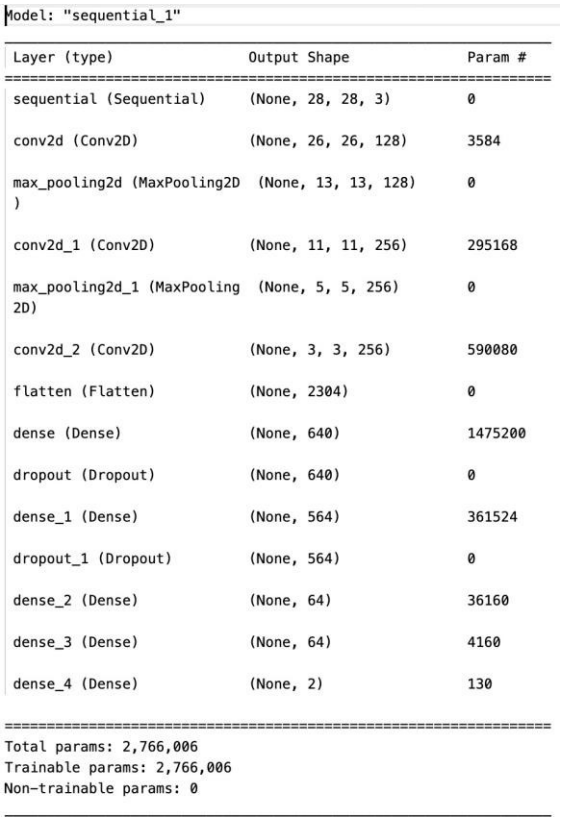


Fig 4.8: loss vs val_loss graph



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential (Sequential)      (None, 28, 28, 3)         0

conv2d (Conv2D)              (None, 26, 26, 128)       3584

max_pooling2d (MaxPooling2D  (None, 13, 13, 128)       0
)

conv2d_1 (Conv2D)            (None, 11, 11, 256)       295168

max_pooling2d_1 (MaxPooling  (None, 5, 5, 256)         0
2D)

conv2d_2 (Conv2D)            (None, 3, 3, 256)         590080

flatten (Flatten)            (None, 2304)              0

dense (Dense)                (None, 640)               1475200

dropout (Dropout)            (None, 640)               0

dense_1 (Dense)              (None, 564)               361524

dropout_1 (Dropout)          (None, 564)               0

dense_2 (Dense)              (None, 64)                36160

dense_3 (Dense)              (None, 64)                4160

dense_4 (Dense)              (None, 2)                 130

=================================================================
Total params: 2,766,006
Trainable params: 2,766,006
Non-trainable params: 0
_________________________________________________________________
```

Fig 4.9: Model summary for second classification model

```
186/186 [==============================] - 38s 125ms/step - loss: 0.6685 - accuracy: 0.5467 - val_loss: 0.4917 - val_accuracy: 0.8019
Epoch 2/10
186/186 [==============================] - 23s 123ms/step - loss: 0.3645 - accuracy: 0.8528 - val_loss: 0.2578 - val_accuracy: 0.9067
Epoch 3/10
186/186 [==============================] - 22s 119ms/step - loss: 0.2942 - accuracy: 0.8839 - val_loss: 0.2255 - val_accuracy: 0.9219
Epoch 4/10
186/186 [==============================] - 22s 119ms/step - loss: 0.2897 - accuracy: 0.8847 - val_loss: 0.2196 - val_accuracy: 0.9210
Epoch 5/10
186/186 [==============================] - 23s 124ms/step - loss: 0.2637 - accuracy: 0.8939 - val_loss: 0.2730 - val_accuracy: 0.8819
Epoch 6/10
186/186 [==============================] - 23s 123ms/step - loss: 0.2529 - accuracy: 0.8995 - val_loss: 0.1920 - val_accuracy: 0.9295
Epoch 7/10
186/186 [==============================] - 24s 128ms/step - loss: 0.2341 - accuracy: 0.9118 - val_loss: 0.2026 - val_accuracy: 0.9229
Epoch 8/10
186/186 [==============================] - 23s 123ms/step - loss: 0.2223 - accuracy: 0.9089 - val_loss: 0.1671 - val_accuracy: 0.9390
Epoch 9/10
186/186 [==============================] - 23s 123ms/step - loss: 0.2164 - accuracy: 0.9161 - val_loss: 0.1643 - val_accuracy: 0.9400
Epoch 10/10
186/186 [==============================] - 23s 122ms/step - loss: 0.2039 - accuracy: 0.9208 - val_loss: 0.2214 - val_accuracy: 0.9257
```

Fig 4.10: accuracy and loss across 10 epochs for second binary classifier

```
test_data = []
image='/Users/utkarsh/majorProject/TB_Chest_Radiograp
img = cv2.imread(str(image))
img = cv2.resize(img, (28,28))
if img.shape[2] ==1:
    img = np.dstack([img, img, img])
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img=np.array(img)
img = img/255
test_data.append(img)

# Convert the list into numpy arrays

test_data1 = np.array(test_data)
```

```
pred=model.predict(np.array(test_data1))
if(np.argmax(pred)):
  print("tuberculosis")
else: print("Normal")
```

```
1/1 [==============================] - 0s 51ms/step
Normal
```

Fig 4.11.: Inference code for second model

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

We have successfully implemented a BS model with an autoencoder which is integrated with a classification model. The BS models generated images that were evaluated by performance metrics methods such as MSE AND MAE. These generated images were further used in the classification model that uses CNN for binary classification which takes as input both the bone suppressed and regular images and distinguishes whether a person has TB or not. We got an accuracy of 95% for the detection of TB from the BS images dataset. Whereas the TB detection from regular chest x-rays was lower with an accuracy of 92% . This lays emphasis on how bone obstruction on the soft tissues of the lungs causes a hindrance in detection and that BS images are better instead for the detection of TB. There are other models for BS image generation with a software on the market, however these will bea costlier alternative. Also these softwares do not have an integrated classification model. Our model has less prerequisites such as requirements of certain softwares, resources or specialized equipment, making it a feasible detection method that can be implemented for small scale detection directly. This is why our model is also cost-effective while providing quick results. Eliminating the dependency on CT Scans or DES systems for initial screening or for diagnosis of certain diseases was our main aim inthis project. This in turn reduces the exposure to radiation and reduces costs by avoiding otherscreening methods. Our model has the potential to make a significant impact in TB detection and diagnosis, and ultimately improve patient outcomes and public health.

The bone suppression model can also be used in other fields of healthcare, such as cardiology and orthopedics. For example, the model could be adapted to remove bone shadows from X-rays of the heart, leading to more accurate diagnosis of heart conditions. Similarly, the model could be used to remove bones from X-rays of joints, improving the diagnosis of musculoskeletal disorders.

REFERENCES

[1] Rajaraman S, Zamzmi G, Folio L, Alderson P, Antani S. “Chest X-ray Bone Suppression for Improving Classification of Tuberculosis-Consistent Findings”. Journal of Diagnostics [Inter-net], MDPI AG, vol. 11, Issue 5 (2021).

[2] Matsubara N, Teramoto A, Saito K, Fujita H. “Bone suppression for chest X-ray image using a convolutional neural filter”. Journal of Australas Physics Engineering Science Medical (2019).

[3] Suzuki, Kenji et al. “Massive training artificial neural network (MTANN) for reduction of false positives in computerized detection of lung nodules in low-dose computed to-mography”. Journalof Medical physics vol. 30,7 (2003).

[4] Chen, Sheng et al. “Enhancement of chest radiographs obtained in the intensive care unit through bone suppression and consistent processing”. Journal of Physics in medicine and bi-ology vol. 61,6 (2016).

[5] Yang, Wei et al. “Cascade of multi-scale convolutional neural networks for bone sup-pression of chest radiographs in gradient domain”. Journal of Medical image analysis vol. 35 (2017)

[6] Rani, Geeta et al. “Spatial feature and resolution maximization GAN for bone suppres-sion in chest radiographs”. Journal of Computer methods and programs in biomedicine vol. 224 (2022).

[7] Li, Feng et al. “Small lung cancers: improved detection by use of bone suppression im-aging--comparison with dual-energy subtraction chest radiography”. Journal of Radiology vol.261,3 (2011).

[8] Zarshenas, Amin et al. “Separation of bones from soft tissue in chest radiographs: Anat-omy-specific orientation-frequency-specific deep neural network convolution”. Journal ofMedical physics vol. 46,5 (2019)

[9] Whitman, Gary J et al. “Dual-energy digital subtraction chest radiography: technical considerations.” Journal of Current problems in diagnostic radiology vol. 31,2 (2002).

[10] Maduskar, Pragnya & Hogeweg, Laurens & Philipsen, Rick & Schalekamp, Steven & Ginneken, Bram. “Improved texture analysis for automatic detection of tuberculosis (TB) on chest radiographs with bone suppression images”. Journal Proceedings of SPIE - The In-ternational Society for Optical Engineering (2013).

[11] Niels Holmark Andersen, Torben Jørgensen, John Brandt Brodersen, “More costs than benefits from screening for cardiovascular disease with computed tomography in the DANCAVAS study”, European Heart Journal, vol. 44, Issue 1, Pages 68–69 (2023).

[12] Power, Stephen P Moloney F, Twomey M, James K, O'Connor OJ, Maher MM. “Com-puted tomography and patient risk: Facts, perceptions and uncertainties”. World journal of radiology vol.8,12 (2016).

[13] Shi, Hai-Min Sun ZC, Ju FH. “Understanding the harm of low-dose computed tomogra-phy radiation to the body (Review)”. Journal of Experimental and therapeutic medicine vol. 24,2 534. (2022).

[14] Minh-Chuong Hyunh. X-ray Bone Shadow Suppression [Internet]. IEEE Dataport (2021).

[15] Gusarev, M., Kuleev, R., Khan, A., Ramirez Rivera, A., & Khattak, A. M.. “Deep learning models for bone suppression in chest radiographs”. 2017 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB) (2017).

[16] Hogeweg, L. E., Mol, C., de Jong, P. A., & van Ginneken, B.. “Rib suppression in chest radiographs to improve classification of textural abnormalities”. Journal of Medical Imaging 2010: Computer-Aided Diagnosis (2010).

[17] T. Rahman et al., "Reliable Tuberculosis Detection Using Chest X-Ray With Deep Learning, Segmentation and Visualization," in IEEE Access, vol. 8, pp. 191586-191601, (2020)

[18] Malik, H., Anees, T., Din, M. *et al."* CDC_Net: multi-classification convolutional neural network model for detection of COVID-19, pneumothorax, pneumonia, lung Cancer, and tuberculosis using chest X-rays". Multimed Tools Applications vol. 82 (2023).

[19] C. Lin et al., "Deep Feature Disentanglement Learning for Bone Suppression in Chest Radiographs," 2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI), Iowa City, IA, USA, pp. 795-798, (2020).

APPENDIX

6.1 AUTOENCODER

Data Preprocessing

```
no_of_images = 4080
img_size = (128,128)
imgs_source = []
imgs_target = []

dir_source = "DS/original"
dir_target = "DS/bone_suppressed"

i = 0
for _, _, filenames in os.walk('DS/original'):
    for filename in filenames:
        print(i)
        i = i+1
        if(i > no_of_images):
            break
        img_source = cv2.imread(os.path.join(dir_source,filename),cv2.IMREAD_GRAYSCALE)
        img_target = cv2.imread(os.path.join(dir_target, filename),cv2.IMREAD_GRAYSCALE)
        # resizing images
        img_source = cv2.resize(img_source,img_size)
        img_target = cv2.resize(img_target,img_size)
    #     # normalizing images
        img_source = np.array(img_source)/255
        img_target = np.array(img_target)/255

        imgs_source.append(img_source)
        imgs_target.append(img_target)
```

4]

Visualising Dataset

```
plt.figure(figsize=(15,10))

for i in range(3):
    ax = plt.subplot(2, 3, i+1)
    plt.imshow(imgs_source[i])
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    plt.title('Source')

    ax = plt.subplot(2, 3, i+4)
    plt.imshow(imgs_target[i])
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    plt.title('Target')
plt.show()
```

Splitting dataset

```
img_rows = 256
img_cols = 256
img_channels = 1
img_shape = (img_rows, img_cols, img_channels)

source = np.array(imgs_source).reshape(-1, img_rows, img_cols, img_channels)
target = np.array(imgs_target).reshape(-1, img_rows, img_cols, img_channels)

source_train, source_test, target_train, target_test = train_test_split(source, target,
                                                    test_size=0.20,
                                                    random_state=1)
```

```
print(source_train.shape, source_test.shape, target_train.shape, target_test.shape)
```

```
(3264, 256, 256, 1) (816, 256, 256, 1) (3264, 256, 256, 1) (816, 256, 256, 1)
```

AutoEncoder model

```
def autoencoder(input_img):
    #encoder
    conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)
    conv1 = BatchNormalization()(conv1)
    conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv1)
    conv1 = BatchNormalization()(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(pool1)
    conv2 = BatchNormalization()(conv2)
    conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv2)
    conv2 = BatchNormalization()(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(pool2)
    conv3 = BatchNormalization()(conv3)
    conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv3)
    conv3 = BatchNormalization()(conv3)


    #decoder
    conv4 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv3)
    conv4 = BatchNormalization()(conv4)
    conv4 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv4)
    conv4 = BatchNormalization()(conv4)
    up1 = UpSampling2D((2,2))(conv4)
    conv5 = Conv2D(32, (3, 3), activation='relu', padding='same')(up1)
    conv5 = BatchNormalization()(conv5)
    conv5 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv5)
    conv5 = BatchNormalization()(conv5)
    up2 = UpSampling2D((2,2))(conv5)
    decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(up2)
    return decoded
```

[22]

Compiling model

```
    input_img = Input(shape = img_shape)
    autoencoder = Model(input_img, autoencoder(input_img))
    autoencoder.compile(loss='mean_squared_error', optimizer = RMSprop() ,metrics=["mae","mse"])
```

```
2023-03-15 00:46:25.934340: I tensorflow/core/common_runtime/pluggable_device/pluggable_device_fac
2023-03-15 00:46:25.934891: I tensorflow/core/common_runtime/pluggable_device/pluggable_device_fac
Metal device set to: Apple M1

systemMemory: 8.00 GB
maxCacheSize: 2.67 GB
```

```
    autoencoder.summary()
```

```
autoencoder.summary()
```

```
Output exceeds the size limit. Open the full output data in a text editor
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 1)]     0

 conv2d (Conv2D)             (None, 256, 256, 32)      320

 batch_normalization (BatchN  (None, 256, 256, 32)     128
 ormalization)

 conv2d_1 (Conv2D)           (None, 256, 256, 32)      9248

 batch_normalization_1 (Batc  (None, 256, 256, 32)     128
 hNormalization)

 max_pooling2d (MaxPooling2D  (None, 128, 128, 32)     0
 )

 conv2d_2 (Conv2D)           (None, 128, 128, 64)      18496

 batch_normalization_2 (Batc  (None, 128, 128, 64)     256
 hNormalization)

 conv2d_3 (Conv2D)           (None, 128, 128, 64)      36928
...
Total params: 427,713
Trainable params: 426,433
Non-trainable params: 1,280
_________________________________________________________________
```

Training Model

```
n_epoch = 50
n_batch = 32
autoencoder_train = autoencoder.fit(source_train, target_train,
                                    epochs = n_epoch,
                                    batch_size = n_batch,
                                    verbose = 1,
                                    validation_data = (source_test, target_test))
```

26]

Checking loss

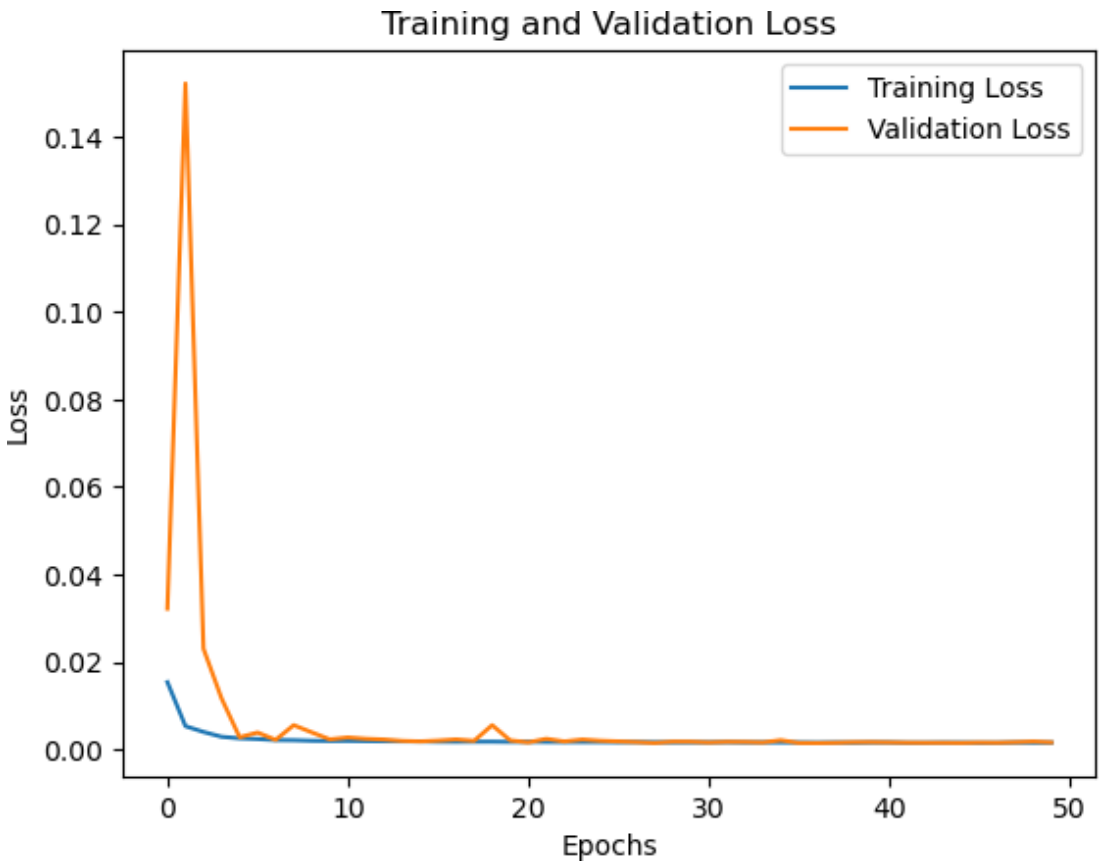
```
interval_epochs = [0, 49]
for e in interval_epochs:
    print("epoch = {}\tLoss = {:.5f}\tValidation_Loss = {:.5f}".format(e+1,autoencoder_train.history[
```

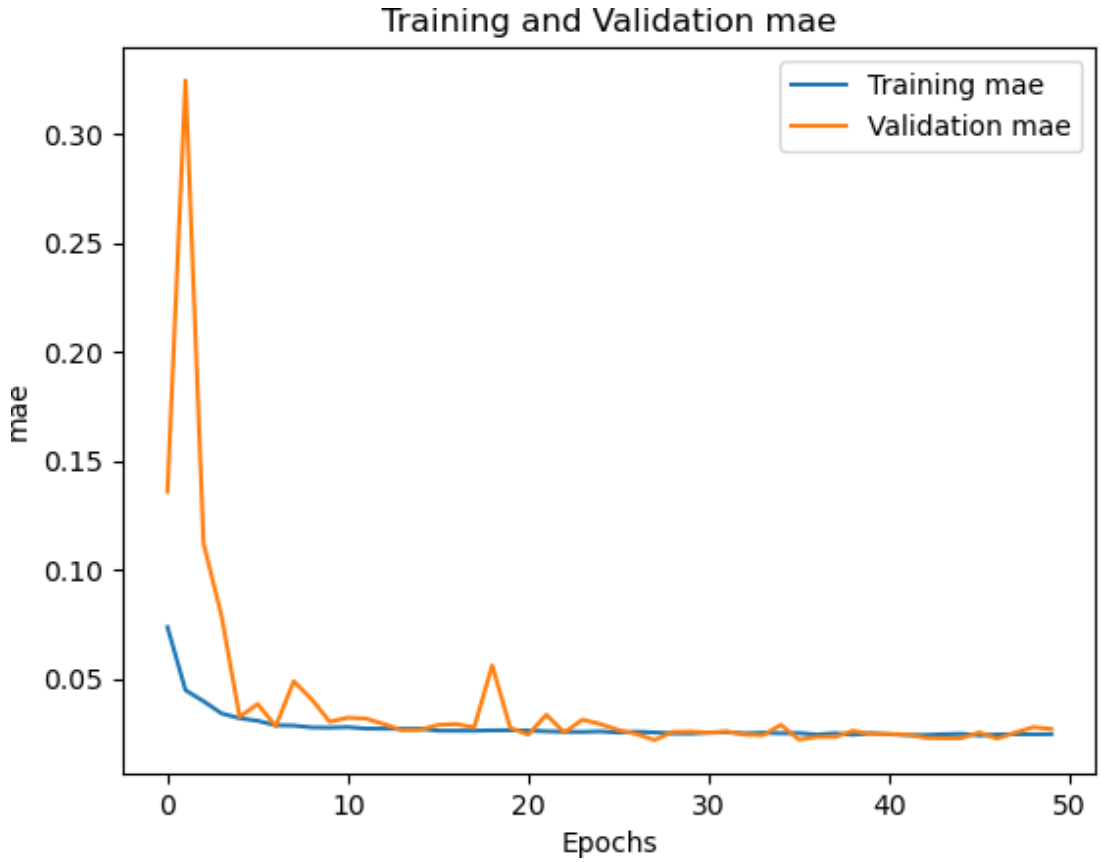
83]

```
epoch = 1       Loss = 0.02059  Validation_Loss = 0.03037
epoch = 50      Loss = 0.00092  Validation_Loss = 0.00124
```

Plotting loss

```
n = np.arange(0, n_epoch)
plt.figure()
plt.plot(n, autoencoder_train.history['loss'], label = 'Training Loss')
plt.plot(n, autoencoder_train.history['val_loss'], label = 'Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```




Training and Validation mae
Training mae
Validation mae
mae
0.30
0.25
0.20
0.15
0.10
0.05
0
10
20
30
40
50
Epochs

6.2 CLASSIFICATION

Reading Data

```
normal_cases_dir = data_dir / 'Normal'
Tuberculosis_cases_dir = data_dir / 'Tuberculosis'
normal_cases = normal_cases_dir.glob('*.png')
Tuberculosis_cases = Tuberculosis_cases_dir.glob('*.png')

train_data = []

for img in normal_cases:
    train_data.append((img,0))

for img in Tuberculosis_cases:
    train_data.append((img, 1))

train_data = pd.DataFrame(train_data, columns=['image', 'label'],index=None)
train_data = train_data.sample(frac=1.).reset_index(drop=True)
train_data.head()
```

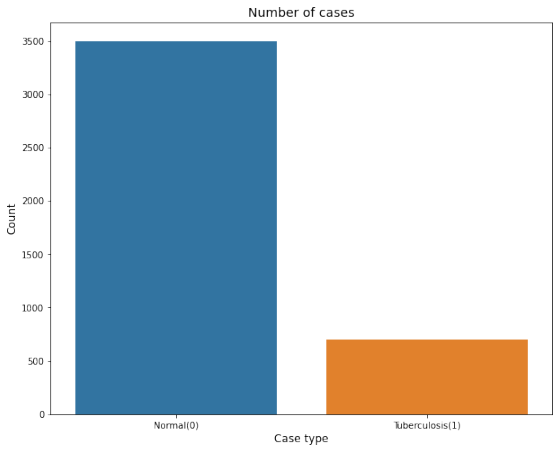
	image	label
0	../input/tuberculosis-tb-chest-xray-dataset/TB...	0
1	../input/tuberculosis-tb-chest-xray-dataset/TB...	0
2	../input/tuberculosis-tb-chest-xray-dataset/TB...	0
3	../input/tuberculosis-tb-chest-xray-dataset/TB...	0
4	../input/tuberculosis-tb-chest-xray-dataset/TB...	0

Counting Data

```
cases_count = train_data['label'].value_counts()
print(cases_count)

plt.figure(figsize=(10,8))
sns.barplot(x=cases_count.index, y= cases_count.values)
plt.title('Number of cases', fontsize=14)
plt.xlabel('Case type', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(range(len(cases_count.index)), ['Normal(0)', 'Tuberculosis(1)'])
plt.show()
```

```
0    3500
1     700
Name: label, dtype: int64
```

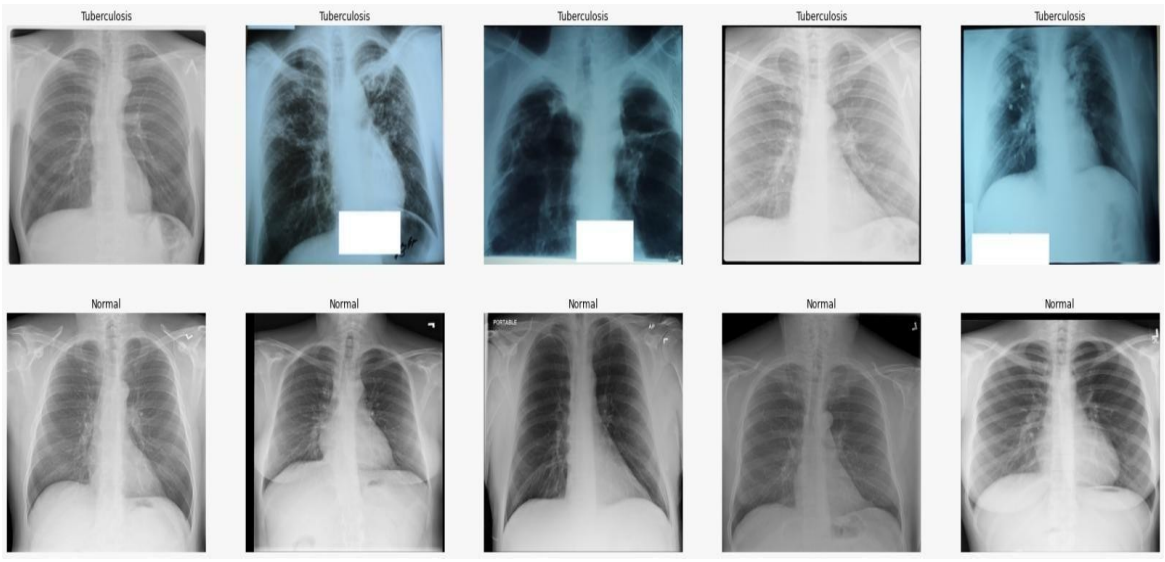


Visualising Dataset

```
Tuberculosis_samples = (train_data[train_data['label']==1]['image'].iloc[:5]).tolist()
normal_samples = (train_data[train_data['label']==0]['image'].iloc[:5]).tolist()

samples = Tuberculosis_samples + normal_samples
del Tuberculosis_samples, normal_samples

# Plot the data
f, ax = plt.subplots(2,5, figsize=(30,10))
for i in range(10):
    img = imread(samples[i])
    ax[i//5, i%5].imshow(img, cmap='gray')
    if i<5:
        ax[i//5, i%5].set_title("Tuberculosis")
    else:
        ax[i//5, i%5].set_title("Normal")
    ax[i//5, i%5].axis('off')
    ax[i//5, i%5].set_aspect('auto')
plt.show()
```



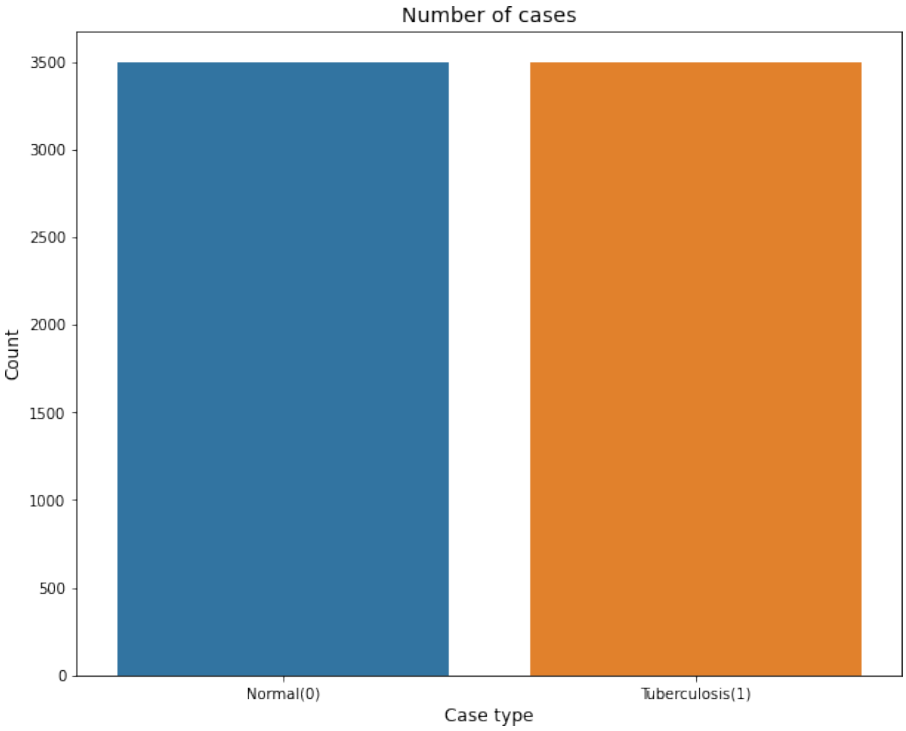
Resampling Dataset

```
from imblearn.over_sampling import SMOTE
smt = SMOTE()
train_rows=len(train_data1)
train_data1 = train_data1.reshape(train_rows,-1)
train_data2, train_labels2 = smt.fit_resample(train_data1, train_labels1)
```

```
cases_count1 = train_labels2['label'].value_counts()
print(cases_count1)

# Plot the results
plt.figure(figsize=(10,8))
sns.barplot(x=cases_count1.index, y= cases_count1.values)
plt.title('Number of cases', fontsize=14)
plt.xlabel('Case type', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(range(len(cases_count.index)), ['Normal(0)', 'Tuberculosis(1)'])
plt.show()
```

```
0    3500
1    3500
Name: label, dtype: int64
```


Number of cases
3500
3000
2500
2000
1500
1000
500
0
Count
Normal(0)
Tuberculosis(1)
Case type

Converting Data

```
from tqdm import tqdm
train_normal = data_dir / 'Normal'
train_Tuberculosis = data_dir / 'Tuberculosis'

# Get the list of all the images
normal_cases = normal_cases_dir.glob('*.png')
Tuberculosis_cases =Tuberculosis_cases_dir.glob('*.png')
train_data = []
train_labels = []
from keras.utils.np_utils import to_categorical



for img in tqdm(normal_cases):
    img = cv2.imread(str(img))
    img = cv2.resize(img, (28,28))
    if img.shape[2] ==1:
        img = np.dstack([img, img, img])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img=np.array(img)
    img = img/255
    label = 'normal'
    train_data.append(img)
    train_labels.append(label)

for img in tqdm(Tuberculosis_cases):
    img = cv2.imread(str(img))
    img = cv2.resize(img, (28,28))
    if img.shape[2] ==1:
        img = np.dstack([img, img, img])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img=np.array(img)
    img = img/255
    label = 'Tuberculosis'
    train_data.append(img)
    train_labels.append(label)


train_data1 = np.array(train_data)
train_labels1 = np.array(train_labels)

print("Total number of validation examples: ", train_data1.shape)
print("Total number of labels:", train_labels1.shape)
```

Creating Model

```
import tensorflow as tf
from tensorflow.keras import  layers, models

model = models.Sequential([
    data_augmentation,
    layers.Conv2D(28, (3, 3), activation='relu', input_shape=(28, 28, 3)) ,
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu')

])

model.summary()
```

```
Model: "sequential_1"
```

```
model.add(layers.Flatten())
model.add(layers.Dense(640, activation='tanh'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(564, activation='tanh'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(64, activation='tanh'))
model.add(layers.Dense(64, activation='sigmoid'))
model.add(layers.Dense(2))
model.summary()
```

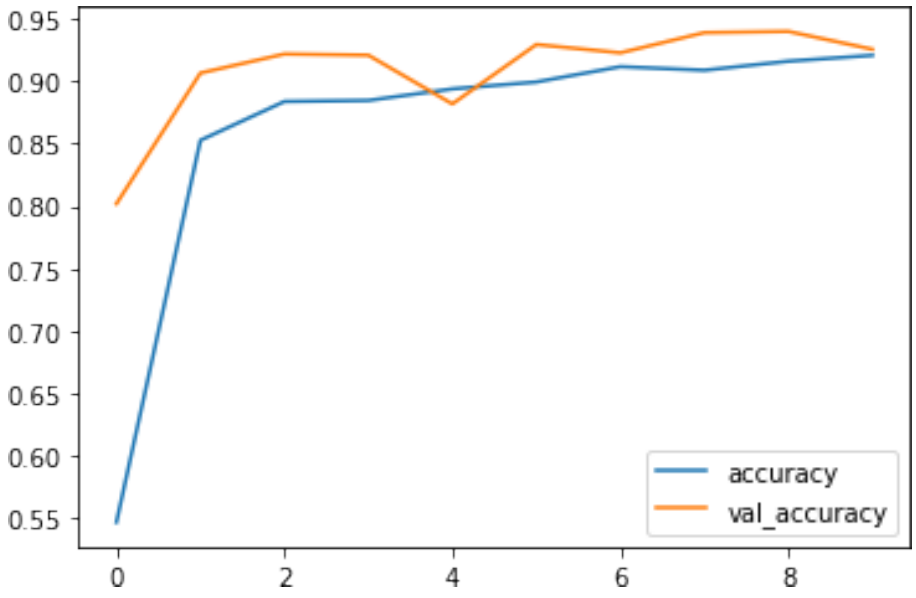
Training

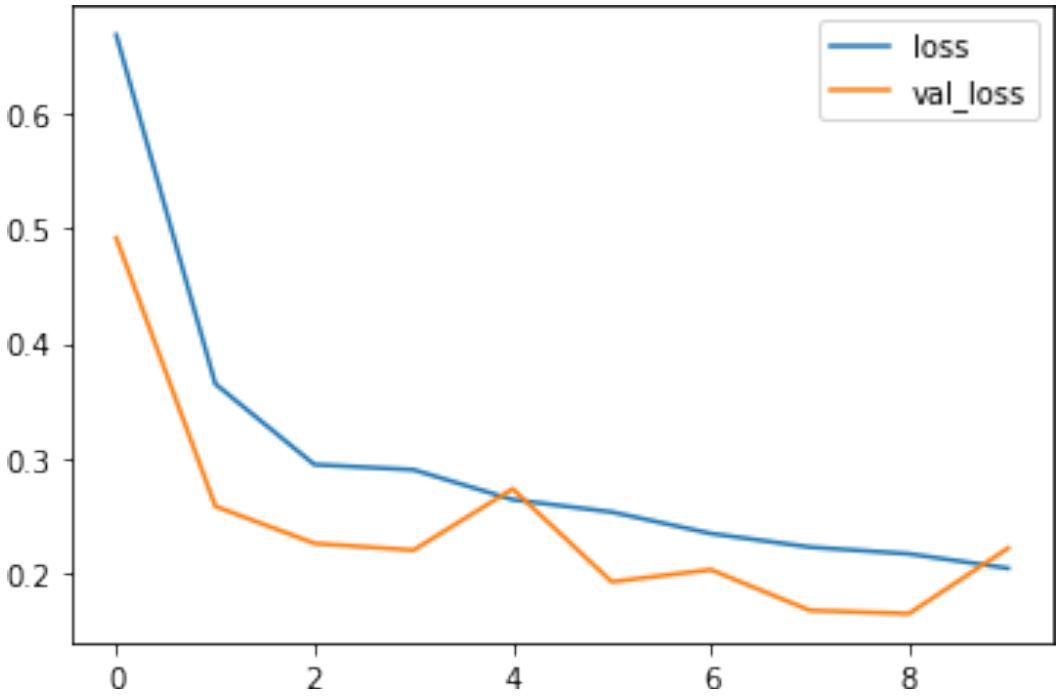
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.fit(np.array(X_train), np.array(y_train), epochs=10, validation_data=(np.array(X_test), np.array(y_test)))
```

```
Epoch 1/10
191/191 [==============================] - 6s 24ms/step - loss: 0.5717 - accuracy: 0.6560 - val_loss: 0.2885 - val_accuracy: 0.9187
Epoch 2/10
191/191 [==============================] - 4s 23ms/step - loss: 0.2583 - accuracy: 0.8977 - val_loss: 0.2445 - val_accuracy: 0.8923
Epoch 3/10
191/191 [==============================] - 4s 22ms/step - loss: 0.2108 - accuracy: 0.9154 - val_loss: 0.1329 - val_accuracy: 0.9451
Epoch 4/10
191/191 [==============================] - 4s 23ms/step - loss: 0.1806 - accuracy: 0.9304 - val_loss: 0.1396 - val_accuracy: 0.9538
Epoch 5/10
191/191 [==============================] - 4s 23ms/step - loss: 0.1528 - accuracy: 0.9452 - val_loss: 0.0758 - val_accuracy: 0.9780
Epoch 6/10
191/191 [==============================] - 5s 27ms/step - loss: 0.1364 - accuracy: 0.9504 - val_loss: 0.0939 - val_accuracy: 0.9703
Epoch 7/10
191/191 [==============================] - 4s 23ms/step - loss: 0.1293 - accuracy: 0.9516 - val_loss: 0.1437 - val_accuracy: 0.9440
Epoch 8/10
191/191 [==============================] - 4s 22ms/step - loss: 0.1069 - accuracy: 0.9650 - val_loss: 0.0705 - val_accuracy: 0.9758
Epoch 9/10
191/191 [==============================] - 4s 23ms/step - loss: 0.1081 - accuracy: 0.9604 - val_loss: 0.0598 - val_accuracy: 0.9813
Epoch 10/10
191/191 [==============================] - 4s 22ms/step - loss: 0.1171 - accuracy: 0.9567 - val_loss: 0.0596 - val_accuracy: 0.9791

<keras.callbacks.History at 0x7f97d8af5790>
```




loss
val_loss
0.6
0.5
0.4
0.3
0.2
0
2
4
6
8

```
test_data = []
image='../input/tuberculosis-tb-chest-xray-dataset/TB_Chest_Radiography_Da
img = cv2.imread(str(image))
img = cv2.resize(img, (28,28))
if img.shape[2] ==1:
    img = np.dstack([img, img, img])
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img=np.array(img)
img = img/255
test_data.append(img)

# Convert the list into numpy arrays

test_data1 = np.array(test_data)
```

```
test_data1.shape
```

```
(1, 28, 28, 3)
```

```
a=model.predict(np.array(test_data1))
a
```

```
array([[-3.1502032,  1.9237326]], dtype=float32)
```

```
np.argmax(a)
```

```
1
```

Loading and Checking Model

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing import image
model = tf.keras.models.load_model('/Users/utkarsh/majorProject/classificationModel')
```

```
test_data = []
image='/Users/utkarsh/majorProject/DS/original/image006.png'
img = cv2.imread(str(image))
img = cv2.resize(img, (28,28))
if img.shape[2] ==1:
    img = np.dstack([img, img, img])
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img=np.array(img)
img = img/255
test_data.append(img)

# Convert the list into numpy arrays

test_data1 = np.array(test_data)
a=model.predict(np.array(test_data1))
if(np.argmax(a)):
    print('Tuberculosis')
else : print('Normal')
```

✓ 0.0s

```
1/1 [==============================] - 0s 13ms/step
Normal
```

6.3 CONVERTING DATASET

Loading bone supression model

```
[3]  from tensorflow import keras
     model = keras.models.load_model('/Users/utkarsh/majorProject/models/256AutoencoderFinal.h5')
     model.summary()
```

Outputs are collapsed ···

Converting Dataset

```
]  #setting paths for conversion
   normal_path = '/Users/utkarsh/majorProject/TB_Chest_Radiography_Database/Normal'
   tuberculosis_path = '/Users/utkarsh/majorProject/TB_Chest_Radiography_Database/Tuberculosis'
```

```
]  img_rows = 256
   img_cols = 256
   img_channels = 1
   img_shape = (img_rows, img_cols, img_channels)
```

Creating image tensor list

```
Tuberculosis_img = []
Normal_img = []
for i in os.listdir(tuberculosis_path):
    image_path = os.path.join(tuberculosis_path,i)
    test1 = []
    test = cv2.imread(image_path,cv2.IMREAD_GRAYSCALE)
    test = cv2.resize(test,(img_rows,img_cols))
    test = np.array(test)/255
    test1.append(test)
    # plt.imshow(test1[0])
    test2 = np.array(test1).reshape(-1, img_rows, img_cols, img_channels)
    Tuberculosis_img.append(test2)

for i in os.listdir(normal_path):
    image_path = os.path.join(normal_path,i)
    test1 = []
    test = cv2.imread(image_path,cv2.IMREAD_GRAYSCALE)
    test = cv2.resize(test,(img_rows,img_cols))
    test = np.array(test)/255
    test1.append(test)
    # plt.imshow(test1[0])
    test2 = np.array(test1).reshape(-1, img_rows, img_cols, img_channels)
    Normal_img.append(test2)

print(len(Tuberculosis_img) , len(Normal_img))
```

13]

```
700 3500
```

Converting Suppressed images to text file

```
for i,file in enumerate(tuberculosis_supressed_final):
    arr_reshaped = file.reshape(x.shape[0], -1)
    np.savetxt("/Users/utkarsh/majorProject/TB/tuberculosis/" + str(i) + ".txt"
```

```
print(x.shape[2], x.shape[2])
```

```
3 3
```

```
loaded_arr = np.loadtxt("/Users/utkarsh/majorProject/TB/tuberculosis/3.txt")
load_original_arr = loaded_arr.reshape(loaded_arr.shape[0], loaded_arr.shape[1]
plt.imshow(load_original_arr,  cmap='gray')
```

```
tuberculosis_supressed = []
for i in range(len(pred_tuberculosis_images)):
    tuberculosis_supressed.append(pred_tuberculosis_images[i][0])
```

```
normal_supressed = []
for i in range(len(pred_normal_images)):
    normal_supressed.append(pred_normal_images[i][0])
```

```
print(len(normal_supressed) , len(tuberculosis_supressed))
print(len(label_normal) , len(label_tuberculosis))
```

```
3500 700
3500 700
```

```
tuberculosis_supressed_final =[]
for i in tuberculosis_supressed:
    img = cv2.cvtColor(i,cv2.COLOR_GRAY2RGB)
    tuberculosis_supressed_final.append(img)
```

```
normal_supressed_final =[]
for i in normal_supressed:
    img = cv2.cvtColor(i,cv2.COLOR_GRAY2RGB)
    normal_supressed_final.append(img)
```

```
print(tuberculosis_supressed_final[0].shape , normal_supressed_final[0].shape)
print(len(tuberculosis_supressed_final) , len(normal_supressed_final))
print(len(label_tuberculosis) ,len(label_normal))
```

```
(256, 256, 3) (256, 256, 3)
700 3500
700 3500
```

```
loaded_arr = np.loadtxt("/Users/utkarsh/majorProject/TB/tuberculosis/3.txt")
load_original_arr = loaded_arr.reshape(loaded_arr.shape[0], loaded_arr.shape[1] // x.shape[2], x.shape[2])
plt.imshow(load_original_arr,  cmap='gray')
```

```
for i,file in enumerate(normal_supressed_final):
    arr_reshaped = file.reshape(x.shape[0], -1)
    np.savetxt("/Users/utkarsh/majorProject/TB/normal/" + str(i) + ".txt", arr_reshaped)
```

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University u/s 3 of UGC Act, 1956)

Office of Controller of Examinations

REPORT FOR PLAGIARISM CHECK ON THE SYNOPSIS/THESIS/DISSERTATION/PROJECT REPORTS

1	Name of the Candidate **(IN BLOCK LETTERS)**	UTKARSH SINHA TANYA ANAND KUMAR	
2	Address of the Candidate	NRE-051 DLF NEW TOWN HEIGHTS, KOLKATA 1005/1006 Orchid Tower, Akurli Road, Mumbai **Mobile Number :** **+91 9874565555** **+91 7509652513**	
3	Registration Number(s)	RA1911003010521	RA1911003010546
4	Date of Birth	03/10/2001	29/11/2001
5	Department	Department of Computing Technologies	
6	Faculty	Engineering and Technology	
7	Title of the Synopsis/Thesis/ Dissertation/Project	Impact of Bone Suppression in Tuberculosis Detection	
8	Name and address of the Supervisor / Guide	SRM Institute of Science and Technology, Kattankulathur **Mail ID :** vijayakc@srmist.edu.in **Mobile Number :** +91 9444328196	
9	Name and address of the Co-Supervisor / Co- Guide (if any)	**Mail ID :** **Mobile Number :**	
10	Software Used	Visual Studio Code, Python, Tensorflow	
11	Date of Verification	11th May 2023	

12	**Plagiarism Details: (to attach the final report)**			
Chapter	**Title of the Chapter**	**Percentage of similarity index (including self-citation)**	**Percentage of similarity index (Excluding self-citation)**	**% of plagiarism after excluding Quotes, Bibliography, etc.,**
1	Introduction	4%	3%	2%
2	Literature Review	4%	4%	3%
3	Proposed Methodology	3%	2%	1%
4	Results and Discussions	1%	<1%	<1%
5	Conclusion and Future Scope	<1%	<1%	<1%
6				
7				
8				
9				
10				
Thesis abstract				
Appendices				

I / We declare that the above information has been verified and found true to the best of my / our knowledge.

Signature of the Candidate(s)	**Signature of the Supervisor / Guide**
Signature of the Co-Supervisor/Co-Guide	**Signature of the HOD**

UtkarshPlagReport2

by Vijayakumaran C

Submission date: 10-May-2023 05:00PM (UTC+0530)
Submission ID: 1744000773
File name: UtkarshReportforPlagChkV2_1Ma23.docx (3.29M)
Word count: 7048
Character count: 38214

UtkarshPlagReport2

ORIGINALITY REPORT

9%	6%	6%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	iopscience.iop.org Internet Source	1%
2	Submitted to Liverpool John Moores University Student Paper	1%
3	Submitted to University of California, Los Angeles Student Paper	1%
4	"Medical Image Computing and Computer Assisted Intervention – MICCAI 2020", Springer Science and Business Media LLC, 2020 Publication	<1%
5	"Computer Vision – ACCV 2018", Springer Science and Business Media LLC, 2019 Publication	<1%
6	thesai.org Internet Source	<1%
7	Submitted to University of Hertfordshire Student Paper	<1%

8	Walter H. L. Pinaya, Andrea Mechelli, João R. Sato. "Using deep autoencoders to identify abnormal brain structural patterns in neuropsychiatric disorders: A large-scale multi-sample study", Human Brain Mapping, 2018 Publication	<1 %
9	www.researchgate.net Internet Source	<1 %
10	webthesis.biblio.polito.it Internet Source	<1 %
11	Chen, S., K. Suzuki, H. MacMahon, and Bram van Ginneken. "", Medical Imaging 2011 Computer-Aided Diagnosis, 2011. Publication	<1 %
12	Chunli Qin, Demin Yao, Yonghong Shi, Zhijian Song. "Computer-aided detection in chest radiography based on artificial intelligence: a survey", BioMedical Engineering OnLine, 2018 Publication	<1 %
13	arxiv.org Internet Source	<1 %
14	Submitted to University of Technology, Sydney Student Paper	<1 %
15	Submitted to United Colleges Group - UCG Student Paper	<1 %

16	www.yourgenome.org Internet Source	<1 %
17	Submitted to Mansfield State High School Student Paper	<1 %
18	mlforsystems.org Internet Source	<1 %
19	mts.intechopen.com Internet Source	<1 %
20	works.bepress.com Internet Source	<1 %
21	Adnane Ait Nasser, Moulay A. Akhloufi. "A Review of Recent Advances in Deep Learning Models for Chest Disease Detection Using Radiography", Diagnostics, 2023 Publication	<1 %
22	Computational Intelligence in Biomedical Imaging, 2014. Publication	<1 %
23	Feng Li, Roger Engelmann, Lorenzo L. Pesce, Kunio Doi, Charles E. Metz, Heber MacMahon. "Small Lung Cancers: Improved Detection by Use of Bone Suppression Imaging—Comparison with Dual-Energy Subtraction Chest Radiography", Radiology, 2011 Publication	<1 %
	dinel.org.uk	

24	Internet Source	<1%
25	Submitted to University of North Texas Student Paper	<1%
26	Submitted to University of Wales Institute, Cardiff Student Paper	<1%
27	ams.confex.com Internet Source	<1%
28	ir.lib.uth.gr Internet Source	<1%
29	Rohit Handa, C. Rama Krishna, Naveen Aggarwal. "Searchable encryption: A survey on privacy ‐ preserving search schemes on encrypted outsourced data", Concurrency and Computation: Practice and Experience, 2019 Publication	<1%
30	Zsolt Szucs-Farkas, Alexander Schick, Jennifer L. Cullmann, Lukas Ebner, Boglarka Megyeri, Peter Vock, Andreas Christe. "Comparison of Dual-Energy Subtraction and Electronic Bone Suppression Combined With Computer-Aided Detection on Chest Radiographs: Effect on Human Observers' Performance in Nodule Detection", American Journal of Roentgenology, 2013 Publication	<1%

31	suzukilab.uchicago.edu Internet Source	<1%

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		