CT LUNG CANCER PREDICTION USING CNN

A PROJECT REPORT

Submitted by

SARAVANA KUMAR M (950421104044)

AROCKIA JERIN J (950421104010)

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

DR.G.U. POPE COLLEGE OF ENGINEERING, SAWYERPURAM-628251

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2025

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"CT LUNG CANCER PREDICTION USING CNN"** is the bonafide work of "**SARAVANA KUMAR M (950421104044), AROCKIA JERIN J (950421104010),**who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported herein 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 on this or any other candidate.

SIGNATURE

Dr. T. Jasperline, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

Professor & Head

Dept of Computer Science And Engineering,

Dr. G U Pope College Of Engineering,

Sawyerpuram-628251.

SIGNATURE

Mrs. S. N Sheebha M.E AP/CSE

SUPERVISOR

Assistant Professor

Dept of Computer Science And Engineering,

Dr. G U Pope College Of Engineering,

Sawyerpuram-628251.

Submitted for Semester Project viva-voce Examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We are grateful our principal **Dr. J. Japhynth, M.E., Ph.D.,** for providing us the environment to develop this project. We are also thankful to Mrs. **Dr. T. Jasperline, M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing the necessary facilitates during the execution of our project work. We also thank for her valuable suggestions, advice, guidance and constructive ideas in each and every step, which was indeed great need towards successful completion of the project. This project would not have been success without our internal guide.

So, we would extend our deep sense of gratitude to our Internal guide **Mrs. S. N SHEEBHA AP/CSE.,** for excellent guidance coordination, motivation and her efforts to bring out the best in us, her availability at all times and thought- provoking questions and timely suggestions. We are very much indebted to our department staff and students for relentlessly supporting us throughout our project work. Finally, our project would have been this shape without lovely efforts from our beloved parents. Their invaluable companionship, warmth, strong faith in our capabilities and they have always helped us to be assertive in difficult times.

ABSTRACT

Lung cancer remains a leading cause of cancer-related mortality worldwide, largely due to the challenges of early detection. Computed Tomography (CT) scans are a standard diagnostic tool for identifying lung abnormalities, such as pulmonary nodules, that may indicate cancer. However, manual analysis of CT images is labor- intensive and prone to subjective interpretation, which can lead to delayed or inaccurate diagnoses.

This project presents a deep learning-based approach for automated lung cancer prediction using Convolutional Neural Networks (CNNs). CNNs are capable of learning hierarchical representations directly from raw image data, making them well-suited for medical image analysis. The proposed system is trained and validated on a publicly available CT scan dataset, and its performance is evaluated using standard metrics such as accuracy, sensitivity, specificity, and AUC-ROC.

The results demonstrate that CNNs can effectively distinguish between cancerous and non-cancerous lung tissue, offering a promising tool to assist radiologists in making faster and more reliable diagnoses. This work contributes to the ongoing efforts in computer-aided diagnosis (CAD) systems and highlights the potential of artificial intelligence to enhance early lung cancer detection and patient outcomes.

Keywords: lung cancer, deep learning, neural networks, convolutional neural networks, CNN, XAI, DenseNet, ResNet, VGG19, transfer learning

PROBLEM STATEMENT

Lung cancer is one of the leading causes of cancer-related deaths worldwide, with early detection being critical for improving survival rates. Computed. Tomography (CT) imaging is a widely used diagnostic tool for identifying pulmonary nodules that may indicate lung cancer. However, manual interpretation of CT scans is time-consuming, prone to inter-observer variability, and requires significant radiological expertise.

This project aims to develop an automated system for lung cancer prediction using Convolutional Neural Networks (CNNs) applied to CT scan data. The goal is to accurately classify CT images as cancerous or non-cancerous, thereby assisting radiologists in early detection and diagnosis. The system will leverage deep learning techniques to learn discriminative features from CT scans and improve diagnostic accuracy, consistency, and efficiency.

This project addresses the problem of lung cancer prediction by developing a CNN-based model that can classify CT scan images into cancerous and non-cancerous categories. The goal is to enhance diagnostic accuracy, reduce the workload on radiologists, and support timely clinical decision-making.

In recent years, artificial intelligence (AI), and in particular deep learning methods such as Convolutional Neural Networks (CNNs), have demonstrated exceptional performance in image classification tasks. CNNs can automatically learn spatial hierarchies of features from large datasets, making them well-suited for analyzing complex medical images like CT scans.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CPU	Central Processing Unit
TPU	Tensor Processing Unit
GPU	Graphics Processing Unit
CNN	Convolution Neural Network
ReLU	Rectified Linear Unit
FC	Fully connected
RGB	Red Green Blue
SVM	Support Vector Machines
PCA	Principal Component Analysis
X-ray	X-ray Radiography
CT	Computed Tomography
XAI	Explainable Artificial Intelligence
CM	Confusion Matrix
ROC	Receiver Operating characteristics
AUC	Area under the ROC
GAN	Generative Adversarial Network
SARSA	State-Action-Reward-State-Action
PET	Positron Emission Tomography

CHAPTER 1

INTRODUCTION

Lung cancer is one of the most fatal diseases and most common causes of cancer deaths worldwide, approximately around 1.80 million in the year 2020 Cancer arises from the transformation of normal cells into tumor cells in a multi-stage process that generally progresses from a pre-cancerous lesion to a malignant tumor. Smoking, alcohol consumption, unhealthy diet, physical inactivity etc. are the common causes of lung cancer. Diagnosing cancer at an early stage increases the chance of performing effective treatment in many tumor groups.

Key approaches include screening patients who are at risk but have no symptoms, and rapidly and appropriately investigating those who do. Machine learning, whereby computers learn complex data patterns to make predictions, has the potential to revolutionize early cancer diagnosis. Such algorithms can assist doctors through analyses of routine health records, medical images, biopsy samples and blood tests to improve risk stratification and early diagnosis. Such tools will be increasingly utilized in the coming years.

Early detection of lung tumor is done by using many imaging techniques such as Computed Tomography (CT), Sputum Cytology, Chest X-ray and Magnetic Resonance Imaging (MRI). Detection means classifying tumor two classes non-cancerous tumor (benign) and cancerous tumor (malignant) .The chance of survival at the advanced stage is less when compared to the treatment and lifestyle to survive cancer therapy when diagnosed at the early stage of the cancer.

Manual analysis and diagnosis system is very hard and more time-consuming for the radiologist. The accuracy of manual detection also depends on the radiologist's years of experience and expertise. So, manual detection and analysis can be greatly improved with the implementation of image processing techniques. It will help to process the data faster and detection accuracy increases in a less period of time.

Neural network plays a key role in the recognition of the cancer cells among the normal tissues, which in turn provides an effective tool for building an assistive AI based cancer detection. The cancer treatment will be effective only when the tumor cells are accurately separated from the normal cells Classification of the tumor cells and training of the neural network forms the basis for the machine learning based cancer diagnosis. CNN algorithm best fit on this application on classification of lung cancer tumor into benign and malignant.

In recent years, deep learning techniques such as convolutional neural networks (CNNs) have been applied to medical imaging for the detection of various diseases, including lung cancer. CNNs are particularly well-suited for this task due to their ability to automatically learn features from images, making them more accurate and efficient than traditional image processing techniques. One potential advantage of using CNNs for lung cancer detection is that they can analyze large amounts of data quickly and accurately, potentially reducing the need for manual review by radiologists. Additionally, CNNs can be trained to recognize patterns and features that may be difficult for humans to discern, potentially improving the accuracy of lung cancer detection.

Overall, the use of CNNs for the detection of lung cancer holds promise as a tool for improving the early detection of this disease, and further research in this area is needed to fully understand the potential benefits and limitations of this approach. The development of a CNN-based lung cancer detection system has the potential to significantly improve the diagnosis and treatment of this disease.

1.1 Background

Every year, cancer is accountable for 9% of deaths, making it the third main source of death from non-communicable diseases and first main source of death from cancer in Nepal [3]. According to GLOBOCAN 2020, there are 11,144 new cases of cancer out of which lung cancer top the list with total 2,505 cases that

is roughly 23% . The rate of cancer incidence and mortality per 100,000 people, adjusted for age, is estimated to be 80.9 and 54.8, respectively. In Nepal, lung cancer is the mostly diagnosed form of cancer in men, representing 18% of new cases and the third most frequent form of cancer in women, making up for 7.7%. Regrettably, the real impact of the disease in terms of incidence, prevalence and mortality is still unknown due to the fact that Nepal did not have a population-based cancer registry until recently.

In Nepal, lung cancer is a major health issue with an augmenting occurrence rate and an elevated mortality rate. According to the data available, lung cancer is one of the

major sources of deaths caused by cancer and its occurrence rate is on the rise, particularly in cities.

There are a number of causes that are responsible for the rising rate of lung cancer in Nepal, such as exposure to environmental pollutants both outdoors and indoors, and lifestyle choices like using tobacco and having an unhealthy diet. Also, limited access to early diagnosis and treatment services and lack of awareness regarding the illness are additional contributors.

Since the mortality rate is quite high and the number of people impacted with lung cancer in Nepal is growing, it is imperative to take effective measures to promote early diagnosis and treatment of the ailment. One promising strategy is the use of Convolutional Neural Networks (CNNs) for lung cancer detection, which could make a considerable difference. Utilization of Convolutional Neural Networks (CNNs) for lung cancer detection is a promising methodology that has the capability to significantly enhance patient outcomes, however, more research and development needs to be done to completely evaluate its workability and efficiency in the Nepalese context.

In addition to technological difficulties, there are also issues concerning

access to medical imaging technology and educated staff, as well as worries about data privacy and safety. Despite these issues, the potential advantages of using CNNs for lung cancer detection in Nepal are considerable, and there is an increasing agreement that this technique has the potential to improve the lives of numerous patients and save lives.

1.2 Domain Introduction

Artificial Intelligence (AI) refers to the imitation of human intelligence in machines that are designed to perform tasks that typically require human cognition in order to perform it. Human cognition includes understanding natural language, recognizing images, making decisions, and solving problems. AI includes various technologies and techniques that include machine learning, deep learning, computer vision, natural language processing and robotics.

Machine learning the one of the subfields of AI where the machines try to learn patterns from the data and use that knowledge to predict output for the unseen input data. It is one of the methods of statistical modeling that enables computers to learn from past data. In machine learning the model is trained by using the huge data present that makes the computer learn the pattern and relationship present on the data. There are various types of machine learning they are:

supervised learning the input data is labeled data and the desired output is known and the model tries to learn the relationship between input and the output data. Examples of supervised learning algorithms are linear regression, decision trees, random forests and support vector machines.

unsupervised learning the data is unlabeled and the model itself tries to find the pattern and relationship on the input data. The unsupervised learning model does not have any target variable to predict

like supervised learning model instead it tries to find the hidden structure of the data, such as clusters, patterns and anomalies. Examples of unsupervised learning includes k-means clustering, principal component analysis (PCA) and hierarchical learning.

semi-supervised learning it combines both supervised and unsupervised learning and is trained on both labeled and unlabeled data. It is used when there is limited labeled data available for training, but also a large amount of unlabeled data is available. Some common semi-supervised learning algorithms include self-training, co-training and generative adversarial networks (GANs).

reinforcement learning an agent tries to learn to make decisions by taking actions in an environment to maximize a reward signal. It gets rewarded if it takes good decision else gets punishment when it makes bad decision as a result agent does not make the same bad decision in future. Here agent is not pre- trained instead it learns from its own experience of the environment. It takes time for an agent to be perfect in its purpose. Some reinforcement learning algorithms include Q-learning, SARSA, and actor-critic methods.

1.3 Objectives

The objectives of this project are divided into main and specific objectives:

1.3.1Main Objectives

The main objective of this project is to design, develop and set up an explainable lung cancer detection system using CNN.

1.3.2 Specific Objectives

- To develop a deep learning-based method for the early detection of lung cancer using CT scan images
- To investigate the potential of using CNNs as a diagnostic tool for lung cancer detection.
- To explore the use of transfer learning techniques to improve the performance of the CNN-based model.
- To ensure interpretability and transparency of the CNN-based model by utilizing explainable AI techniques (SHAP).
- To design and implement web applications that provide interactive interface to the user

CHAPTER 2
LITERATURE REVIEW

[1] Title: Deep Learning for Lung Cancer Detection in CT Images (2018):

Authors: X. Li, Y. Zhao, M. Tang

This study introduced a basic 2D CNN model trained on the LIDC-IDRI dataset for classifying lung nodules as benign or malignant. It demonstrated the CNN's capability to learn hierarchical image features, achieving high sensitivity but facing challenges with false positives due to nodule similarities.

[2] Title: 3D Convolutional Neural Networks for Pulmonary Nodule Detection (2019):

Authors: R. Wang, S. Cheng, L. Yu

This paper employed a **3D CNN** to exploit volumetric information in CT scans, which enhanced spatial feature learning. Compared to 2D CNNs, this approach provided improved accuracy in distinguishing malignant nodules, but at the cost of higher computational demand.

[3] Title: Transfer Learning for Lung Cancer Prediction using Pretrained CNNs (2020):

Authors: M. Singh, D. Patel

The authors used pretrained CNN models like VGG16 and ResNet50, fine-tuned on CT datasets. Transfer learning significantly reduced training time and data dependency while maintaining high classification accuracy. However, adapting pretrained models to medical imaging required careful tuning.

[4] Title: Hybrid Deep Learning Framework for Lung Nodule Classification (2021):

Authors: J. Rao, T. Lin

This paper proposed a hybrid model combining CNNs with radiomic feature extraction and decision-level fusion. The system improved diagnostic

performance and interpretability by leveraging both deep and handcrafted features.

[5] Title: Attention-Based CNN for Early Lung Cancer Detection (2022):

Authors: L. Zhang, H. Chen

The study introduced attention modules within CNN layers to focus on salient lung regions. This improved the model's ability to localize and classify nodules, reducing false positives and increasing clinical trustworthiness.

[6] Title: Multi-Task CNNs for Joint Segmentation and Classification of Lung Nodules (2022):

Authors: P. Verma, R. Shah

A multi-task CNN was developed to perform both nodule segmentation and malignancy prediction simultaneously. The integration of segmentation improved contextual understanding, thereby boosting classification performance.

[7] Title: Lightweight CNNs for Lung Cancer Screening in Low-Resource Settings (2023):

Authors: A. Desai, S. Kumar

Focused on deploying lung cancer models in resource-constrained environments, this research designed lightweight CNNs with pruning and quantization. Although slightly less accurate, these models were computationally efficient and suitable for portable CT screening units.

CHAPTER 3
SYSTEM ANALYSIS

3.1 Existing System

The existing systems for lung cancer detection using Computed Tomography (CT) scans typically rely on conventional image processing techniques and early-stage machine learning models. These systems generally involve manual or semi-automated feature extraction processes, where radiologists or algorithms identify nodules based on shape, size, and texture features. Traditional classifiers such as Support Vector Machines (SVM), Random Forests, and Decision Trees have been used to categorize these features into benign or malignant classes.

However, these conventional methods often suffer from limited accuracy and generalizability due to their reliance on handcrafted features, which may not capture the complex patterns and variations present in lung nodules. Moreover, manual analysis is time-consuming, prone to human error, and not scalable for large datasets.

Recent systems have attempted to integrate Computer-Aided Detection (CAD) tools to assist radiologists in identifying potential cancerous lesions. While these tools have improved diagnostic throughput, they still face challenges in sensitivity and specificity, particularly when distinguishing between benign abnormalities and early-stage malignancies.

The emergence of deep learning, particularly Convolutional Neural Networks (CNNs), has begun to address some of these limitations. However, in many existing systems, CNNs are either not fully optimized or lack sufficient annotated datasets, which restricts their effectiveness. Furthermore, some models focus only on nodule classification, neglecting the critical stages of detection and segmentation, which are crucial for comprehensive analysis and diagnosis.

3.2 Proposed Sysytem

The proposed system aims to enhance the accuracy, efficiency, and reliability of lung cancer prediction by leveraging Convolutional Neural Networks (CNNs) for automated analysis of CT scan images. Unlike traditional systems that rely heavily on manual feature extraction and classical machine learning techniques, this system utilizes a deep learning approach that can automatically learn hierarchical feature representations directly from raw image data.

The architecture of the proposed system is designed to perform the following key tasks:

- **Preprocessing**: CT scan images are preprocessed to normalize intensity values, remove noise, and standardize image dimensions. Techniques such as lung segmentation, histogram equalization, and resampling are applied to improve image quality and consistency.
- **Nodule Detection and Segmentation**: Advanced image processing and CNN-based models are used to detect and segment potential nodules in the lung regions. Accurate localization and segmentation of nodules are critical for reducing false positives and improving classification performance.
- **Feature Extraction and Classification**: A CNN model is employed to extract deep features from the segmented regions. These features capture complex patterns associated with malignancy, including texture, shape, and contextual information. The model then classifies nodules as benign or malignant with high precision.

CHAPTER 4
INTRODUCTION TO CONVOLUTIONAL NEURAL NETWORK

A Convolutional Neural Network (CNN) is a type of deep literacy neural network that's generally used for image and videotape recognition tasks. It's specifically designed to reuse grid- suchlike data, similar as an image, and it uses convolutional layers to prize features from the input. The introductory structure block of a CNN is the convolutional subcaste, which performs a fine operation called complication. The complication operation involves applying a set of pollutants, also called kernels or weights, to the input data, which produces a point chart.

Each sludge is designed to describe a specific point in the input data, similar as edges or textures. By mounding multiple convolutional layers, the CNN can learn decreasingly complex features of the input data, similar as shapes and objects. Between the convolutional layers, a pooling subcaste is frequently used, which reduces the spatial confines of the point chart, making the model more robust to small restatements of the input and reduce the number of parameters. Eventually, the affair of the CNN is passed through a completely connected subcaste, which produces the final affair of the model, similar as a class marker.

CNNs have been veritably successful in image and videotape recognition tasks, similar as object discovery, image bracket, and semantic segmentation. They're also used in natural language processing tasks similar as textbook brackets.

Example:

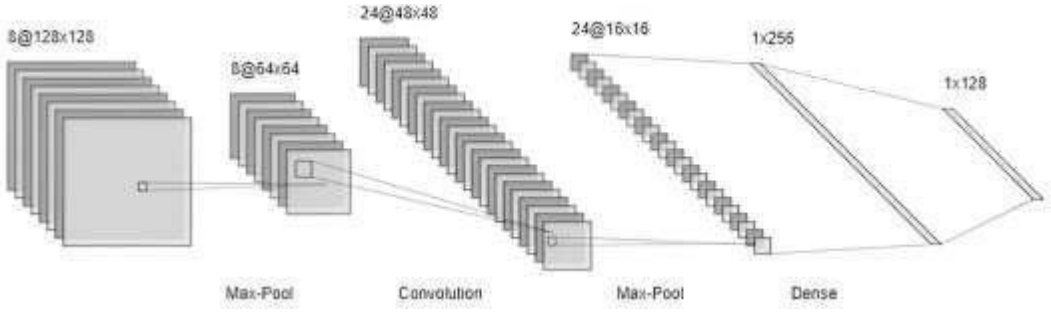


Figure 4.1 Architecture of CNN

Figure shows the basic architecture of Convolutional neural network, now driving towards the mathematical intuition behind CNN architecture. There are four main operations in the ConvNet as shown in Figure 3 above:

1. Convolution
2. Non-Linearity (ReLU)
3. Pooling or Sub Sampling
4. Classification (Fully Connected Layer)

4.1 Convolution

In a Convolution there are some terminologies that are used they are kernel, padding and stride. A kernel, also known as a filter or a weight, is a small matrix of numbers that is used in the convolution operation to extract features from the input data. Each kernel is designed to detect a specific feature in the input data, such as edges, textures, or patterns. Padding is a technique used in convolutional neural networks (CNNs) to control the spatial size of the output feature maps. When a kernel is convolved with the input data, it slides over the input data, elementwise multiplies the values at the overlapping positions and sums them up, resulting in a new value. This process, however, reduces the spatial size of the feature map. If the input image is large, this reduction in spatial size may be significant, and the network may lose important information. Stride is a hyper parameter used in convolutional neural networks (CNNs) to control the spatial size of the output feature maps and to reduce the

computational cost. When a kernel is convolved with the input data; it slides over the input data with a fixed step size, called the stride. The stride is the number of pixels by which the kernel moves in the horizontal and vertical directions as it slides over the input data. A larger stride will result in the kernel covering less of the input data, which in turn will result in a smaller output feature map. The convolution between the input image and kernel is given by the formula:

$$T(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

Here "I" is the input matrix of image "K" is the filter or kernel matrix and "T" is the resultant matrix after convolution between matrix and kernel. Let us consider I as 6x6 image and taking K as 3x3 filter as shown below:

$$I = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \qquad K = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

When a kernel is convolved with the input data, it slides over the input data, elementwise multiplies the values at the overlapping positions and sums them up, resulting in a new value. This process produces a feature map that highlights the presence of the specific feature that the kernel was designed to detect. The result of convolution of I and k is given as:

$$T = (I * k) = \begin{bmatrix} 0 & -4 & -4 & 0 \\ 0 & -4 & -4 & 0 \\ 0 & -4 & -4 & 0 \\ 0 & -4 & -4 & 0 \end{bmatrix}$$

Here the final output is scaled using scalar and the necessary feature information

is extracted. We can apply as many filters as we want in convolution layers that go through the same operation as above. The final output shape of the resultant matrix is determined by following formula:

$$\frac{(w - f + 2p)}{S} + 1$$

4.2 Non-Linearity (ReLU):

Non-Linearity is an essential component of neural networks, especially deep learning models, because it allows the model to learn complex and non- linear relationships between the input and output data. The Rectified Linear Unit (ReLU [16]) is a commonly used non-linear activation function in deep learning.

The ReLU function is defined as f(x) = max(0,x), where x is the input to the function. It is a simple function that returns 0 if the input is negative and returns the input if the input is positive. In other words, it clips all negative values to zero, and it passes the positive values through unchanged.

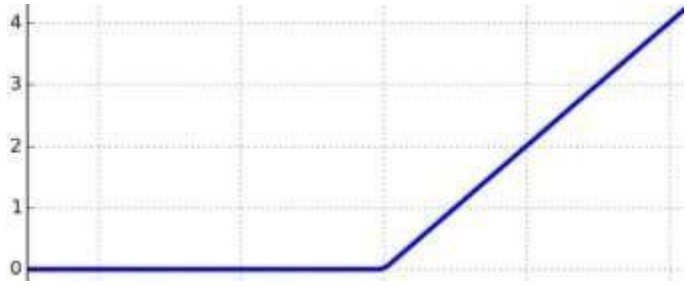


Figure 4.2 ReLU activation function curve

The ReLU activation function has various advantages compared to other activation functions, like sigmoid and tanh, such as being computationally efficient since it only needs a straightforward comparison operation. It also helps to solve the problem of vanishing gradients, which is a common difficulty in deep neural networks. This problem shows up when the gradients of the weights become very small as the error is backpropagated through the layers, making it

tough for the network to learn. ReLU activation functions can fight this problem because their gradients are either 0 or 1.

Furthermore, there are several variations of the ReLU function, like Leaky ReLU, Parametric ReLU, and Exponential Linear Unit (ELU) that are used to address the issue of dying ReLU, where the ReLU function produces an output of zero for all negative input. ReLU activation functions are widely used in deep learning and are implemented in the hidden layers of the neural network to introduce non-linearity.

4.3 Pooling or Sub-sampling

Pooling or sub-sampling is one of the important methods used in convolutional neural networks (CNNs) to reduce the spatial size of the feature maps and makes the computational cost of the neural network more efficient.

The pooling layer is used after the convolutional layers of the CNN, and it is applied to down-sample the feature maps. There exist various pooling operations that could be used but the most used pooling operation is max pooling, which selects the maximum value from a defined region of the feature map, called the pooling window or kernel, and replaces the entire region with that one maximum value. This process helps in reducing the spatial size of the feature map by a factor of the pooling window size.

By reducing the spatial size of the feature map, pooling can make CNN more robust to small translations of the input and reduce the number of parameters. It also helps to reduce overfitting by reducing the number of parameters in the network. The value of stride for Pooling operation can be set greater than 1 so that we can further reduce the spatial size of the feature map and it can be applied to both the width and the height of the feature map.

In addition to max pooling, there are other also several types of pooling

operations such as average pooling, which selects the average value from a small region of the feature map, sum pooling, which selects the sum of the values from a small region of the feature map and many more.

For example, in "T" matrix, we can apply max-pooling and form another matrix of T':

$$T = \begin{bmatrix} 10 & 8 & 4 & 9 \\ 3 & 4 & 9 & 10 \\ 3 & 7 & 7 & 9 \\ 5 & 4 & 12 & 8 \end{bmatrix}$$

After applying max pooling of window size of 2 and stride of 2 we get the resultant matrix as:

$$T' = \begin{bmatrix} 10 & 10 \\ 7 & 12 \end{bmatrix}$$

4.4 Classification (Fully Connected Layer):

The last step in a convolutional neural network (CNN) is the classification step, which is performed by using a fully connected layer which is also known as classification layer in CNN.

A fully connected layer, also known as a dense layer, is a flattened version of the convolutional layer. It is called fully connected because all the neurons in the previous layer are connected to all the neurons in the current layer and forms a computational network.

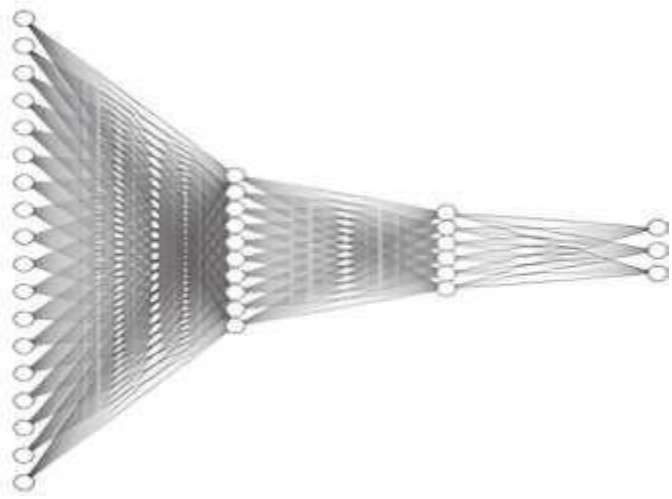


Figure 4.3 Fully-connected Neural Network

The final output of the convolutional and pooling layers is passed through a fully connected layer, which maps the output of the previous layers to the final output of the network. The number of neurons in the last part of the fully connected layer is usually equal to the number of classes in the dataset. The fully connected layer uses a set of weights and biases, which are learned by the model, while the training process in order to make the final prediction. Therefore, the output of the final fully connected layers gives the probability of the belongings of the particular input image among the different classes for which the model is initially trained which is done using a SoftMax function for multiclass classification problem.

The fully connected layer is also where the dropout technique is applied. This is a regularization technique that is used to prevent overfitting by randomly dropping out some neurons during each forward pass of the input features through the layer. This reduces the complexity of the model and forces the remaining neurons to learn a more robust representation of the input data. It is one of the hyper parameters of the model that needs to be set during model definition.

As mentioned earlier at the last section of the CNN layers we use the SoftMax function [17] to test the result that belongs to which particular class. The SoftMax function is commonly used in multiclass classification problems, where the goal is to predict one of several possible outcomes. For example, in image classification, the goal may be to predict which of several objects are present in an image, such as a dog, cat, or car. In this example, the SoftMax function can be used to convert the output of a neural network into a probability distribution over the possible classes and whichever class has the highest probability our model result will be inclined towards that particular class.

The SoftMax function is defined as follows:

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_{i=i}^{N} e^{x_i}}$$

Where x is a vector of input values, e is the base of the natural logarithm, and N is the number of output classes. The numerator of the SoftMax function calculates the exponential of each input value, while the denominator calculates the sum of the exponentials of all input values. This results in a set of values between 0 and 1, representing the probabilities of each class.

4.5 Confusion Matrix and model evaluation Metrics

Confusion matrix is a simple way to lay out how many predicted categories or classes were correctly predicted and how many were not. It is used to evaluate the result of a predictive model with a class outcome to see the number of classes that were correctly predicted as their true class. In matrix form, it can be expressed as:

$$Confusion\ Matrix = \begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$$

For Example:

N=165	**Predicted: Positive**	**Predicted: Negative**	
Actual Positive:	TP = 100	FN = 5	105
Actual Negative	FP = 10	TN = 50	60
	110	55	

Table 4.1 Confusion Matrix

CHAPTER 5
PROJECT METHODOLOGY

5.1 Block diagram of the system

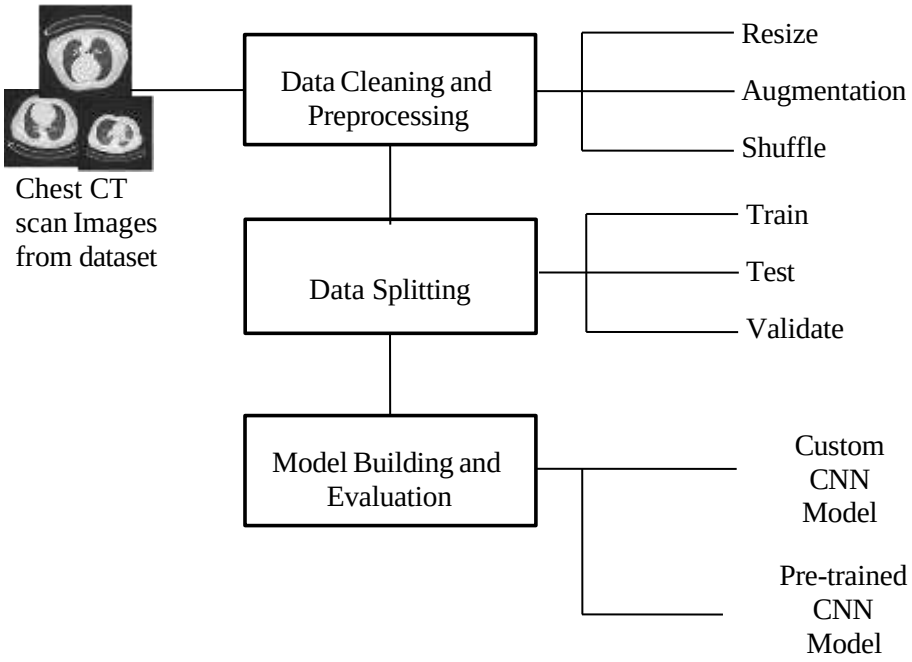


Figure 5.1 Block diagram of Model Building and Evaluation

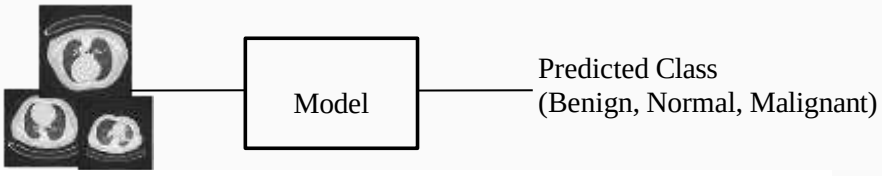


Figure 5.2 ***Prediction using model.***

First of all, we will collect the labeled dataset from the trusted media and then after we will apply some preprocessing steps to the collected CT scan images. After that, the preprocessed data will be split into training, testing and validation sets for training, testing and validation purposes respectively. Then we will train the models with the data and evaluate them on the basis of performance metrics and compare the results.

5.2 Data collection and data introduction

The dataset is taken from the Iraq-Oncology Teaching Hospital/National Center for Cancer Diseases (IQ-OTH/NCCD). The dataset is available online and the link of the website is https://data.mendeley.com/datasets/bhmdr45bh2/1. This dataset was collected in this hospital over a period of three months in fall 2019. The dataset includes CT scans of patients diagnosed with lung cancer in different stages, as well as healthy subjects. This dataset contains a total of 1197 CT scans images representing three classes such as normal, Benign, and Malignant. Here normal categories of CT scans represent the CT scans that do not contain any cancerous cells or tumor. The Benign categories of CT scans contains the CT scans that contain benign cells or tumors which are not cancerous and may behave exactly like normal cells. They may grow more slowly or faster than normal cells, but they do not spread to other parts of the body and the tumor takes a more regular shape. Malignant categories of the CT scans contain cells and tumors which are cancerous and can spread to other parts of the body. They often grow and divide more rapidly than normal cells and do not respond to normal body signals to stop dividing and tumor takes more irregular shapes.

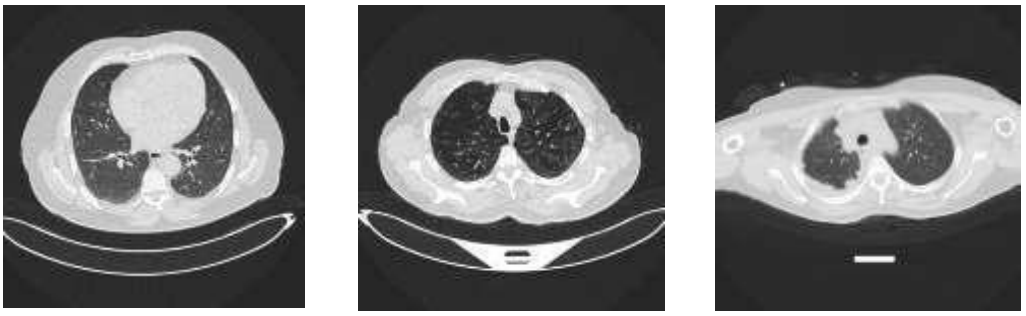


Figure 5.3 Sample of dataset belonging to class Benign, Normal and Malignant

The dataset contains a total of 416 Normal cases CT scans, 120 benign cases CT scans and 561 malignant cases CT scans. The dataset seems to be imbalanced, so the proper steps were taken to balance the dataset on the following next step.

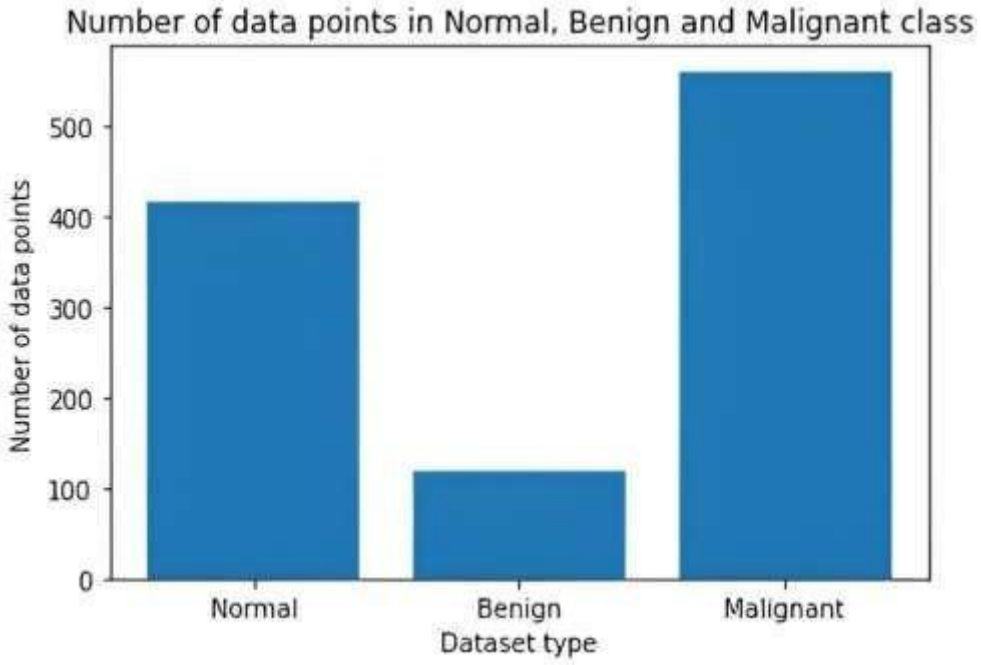


Figure 5.4 The distribution of the dataset among different classes

5.3 Data pre-processing:

The set of pre-processing steps were taken to make the dataset balance and ready for training the model. First, it is necessary to perform the data augmentation operation. In data augmentation processes first of all the CT scans images were resized to 246X256, and then random rotation is performed with the 15 degrees to provide variation in the datasets. After random rotation center crop operation was performed with the same image size and some random horizontal flip was performed. Finally, the image was converted into tensor, and it is normalized by this value ([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]).

Since by observation of the distribution of the dataset it is clearly imbalanced dataset, so we need to perform certain operation to balance it otherwise it may result in several disadvantages such as:

Bias towards the majority class: If the dataset has a skewed distribution like we observe earlier then the model may be biased towards the majority class.

Poor performance on minority class: In an imbalanced dataset, the model may

result in poor performance towards minority class since the minority class does not have enough examples to learn form and may lead to misclassification and false negatives.

Inadequate evaluation metrics: In an imbalanced dataset, accuracy is not the inadequate evaluation metrics because the model may achieve high accuracy simple by always predicting the majority class. So it is essential to evaluate other metrics, such as precision, recall, and F1-score as well.

Hence in order to balance the dataset there are various mechanisms among which Cost-sensitive learning is used. In cost-sensitive learning the misclassification cost are assigned to each class where the minority class is given more weight which eventually help the model to focus more on the correctly classifying the minority class. For example, in a medical diagnosis problem, misclassifying a cancer sample as healthy may have a higher cost than misclassifying a healthy sample as cancer.

5.4 Model Selection and Model Design

CNN model is preferred instead of normal feed-forward neural network for this application because of following main reasons:

CNNs are specialized for structured data as it preserves the spatial structure of the image for example, they transform a 2D input into a 2D output which is a transformed image.

In feed-forward network, first unwrap of the 2D input image into a 1D vector (rows of pixels) and hence the spatial information is lost in the process.

In feed-forward network the pixels which are close to each other in the original image are now far apart. Also, that were separated appears next to each other in the vector.

In this application, a custom CNN architecture is built along with the various pre- trained models are also used to train on the same dataset so that we can better choose the best performing model and also compare their efficiency.

5.4.1 Custom CNN model:

A custom CNN architecture is a type of deep learning model that is designed and trained for a particular task. The basic building blocks of a custom CNN are its layers,

which include various convolutional layers, pooling layers, activation layers, fully connected layers, normalization layers and dropout layers. In the custom CNN architecture number and types of layers and various hyperparameters such as number of filters, kernel size and stride size.

We designed our own custom CNN model that best fits our problem after performing various hyperparameter tuning. The various layers and associated trainable parameters of the custom shown CNN model architecture is listed on a table below. According to which our total model parameters are 154,851,369 among which total trainable parameters are 154,851,369 and total non-trainable parameter is zero.

Layer	Output Shape	Kernal Shape	#params	#(weights + bias)	requires_grad
Conv2d-1	[1, 6, 256, 256]	[6, 3, 3, 3]	168	(162 + 6)	True True
BatchNorm2d-2	[1, 6, 256, 256]	[6]	12	(6 + 6)	True True
ReLU-3	[1, 6, 256, 256]				
MaxPool2d-4	[1, 6, 128, 128]				
Conv2d-5	[1, 12, 128, 128]	[12, 6, 3, 3]	660	(648 + 12)	True True
BatchNorm2d-6	[1, 12, 128, 128]	[12]	24	(12 + 12)	True True
ReLU-7	[1, 12, 128, 128]				

MaxPool2d-8	[1, 12, 64, 64]				
Conv2d-9	[1, 32, 64, 64]	[32, 12, 3, 3]	3488	(3456 + 32)	True True
BatchNorm2d-10	[1, 32, 64, 64]	[32]	64	(32 + 32)	True True
ReLU-11	[1, 32, 64, 64]				
MaxPool2d-12	[1, 32, 32, 32]				
Flatten-13	[1, 32768]				
Linear-14	[1, 5461]	[5461, 32768]	178951509	(178946048 + 5461)	True True
Dropout-15	[1, 5461]				
Linear-16	[1, 3]	[3, 5461]	16386	(16383 + 3)	True True

Table 5.1 Model architecture and relate parameters

5.4.2 Transfer learning:

A pre-trained CNN model [20] is a model that has already been trained a very huge datasets belonging to the large number of classes and their learned parameters are saved which we can use latter for our own purpose after doing certain changes on the model architecture which is also known as fine-tuning. This whole process is known as transfer learning as we used already trained model to our own purpose after making certain changes according to our need. Pre-trained CNN models are often made available by research organizations and deep learning libraries after their research and model development. In many cases using pre-trained models can greatly speed up the process of training a new model and can lead to improved performance compared to training a model from scratch, particularly when the new task has limited labeled data. For our task we tried about three pre-trained model for transfer learning purposes they are as follows:

- DenseNet121
- ResNet152

- VGG19

DenseNet Model: In order to compare the performance of the model, we implemented our problem with DenseNet model having 121 layers in it. DenseNet model architecture is introduced in the paper "Densely Connected Convolutional Networks" published in 2017 [21]. This model was developed to address the problem of vanishing gradients in deep neural networks. The key idea behind the DenseNet is to concentrate the feature maps from each layer in the network, instead of using element-wise addition as in traditional Residual Networks. This allows the network to pass information from earlier layers to later layers more efficiently, reducing the risk of vanishing gradient problem and improving the overall performance of the model.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Table 5.2 DenseNet model architecture

In order to fine-tune this model to fit on our problem domain we performed some changes in the classifier layer of the DenseNet121 model. First of all we added a linear layer on it which will convert incoming 1024 input features to 128 output features. After that we added dropout layers of 0.2 for the purpose of regularization to reduce overfitting. Similar process is repeated for another linear layer that converts input 1028 features to output 102 features and dropout layer is added with 0.2. Finally a classification linear layer is added that converts input 102 features to 3 output features. After fine-tuning the model, the model has following parameters details:

- Total parameters 7,098,523
- Total Non-Trainable parameters
  6,953,856 Total Trainable parameters
  144,667

Finally, a model graph representation is shown below that shows the connections between layers and the flow of data through the network.

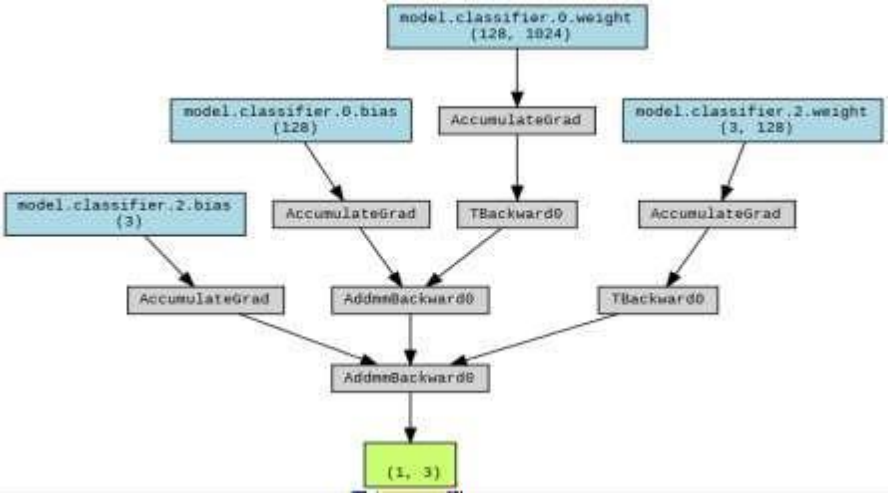


Figure 5.5 DenseNet121 Model flow diagram after fine-tuning

ResNet152 Model: ResNet is another model that we tried to implement in order to visualize how well it performs for our problem domain. ResNet architecture is first introduced in the paper "Deep Residual Learning for Image Recognition" in 2015 The ResNet model was built in order to make the optimization easy and reduce vanishing gradient problems as the number of layers increased in the network. The ResNet model is called ResNet because it uses residual connections, which are connections that bypass one or more layers. ResNet consists of multiple blocks and each block of it consists of multiple convolutional layers, batch normalization layers and activation functions. In this model residual connections are added by adding the input to the output block which eventually allows the network to learn the residual mapping between the input and output rather than trying to learn the entire mapping from the scratch which results in more efficient learning pattern. If we

talk about its application, it is widely being used by various computer vision tasks, such as image classification, object detection and semantic segmentation purposes. There are several versions of ResNet such as ResNet-18, ResNet-34, ResNet-150, ResNet-101, and ResNet-152. We used ResNet-152 as the deeper the network, the greater the performance.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 5.3 ResNet Model Architecture

In order to finetune the model to fit in our problem domain, we changed its fully connected (fc) layer. In fc, first we added one linear layer that accepts input features of 2048 neurons and gives output of 256 neurons. After the linear layer we added one dropout layer of 0.2 that acts for model regularization. The same linear layer and dropout layer is repeated once again which convert input 256 neurons to output 204 neurons. Finally, a linear layer that accepts the input-features of 204 and gives output

features of 3 is added which acts as a classifier. Finally, we get the model with following details:

Total parameters 58,669,123

Total Non-Trainable parameters 58,143,808

Total Trainable parameters 525,315

The resulted model graphical representation when we pass the data through the network is resulted as follow:

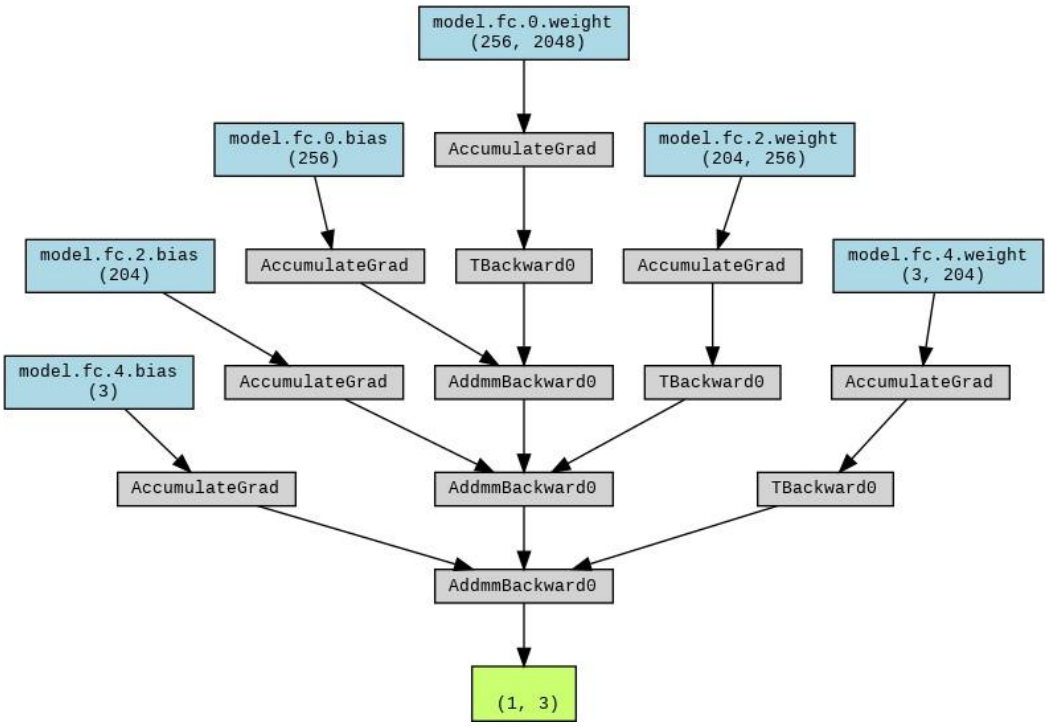


Figure 5.6 ResNet Model flow diagram

VGG19 model: VGG is a convolutional neural network architecture that was developed by Visual Geometry Group (VGG) at the University of Oxford. It was first introduced in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" which was published in 2014 . The VGG model makes use of multiple convolutional layers with various depth. The popular VGG models are VGG- 16 and VGG-19. In our case we used VGG19 as it offers more depth and helps to capture the complex features of our model. Like other pre-trained models, VGG is also trained on large datasets and are optimized during the training process. As a result it is capable of classifying various images of different categories. The popular use cases of VGG are image recognition tasks including object classification, scene recognition,

and fine-grained recognition.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 5.4 VGG model architecture

In order to fine-tune this model, we try to modify its classifier layers as done before on other pre-trained models. We tried to create consistent classifier layers in each pre- trained model such that we added linear layer and dropout layer of 0.2. Again, the same layers are repeated but give output of lesser neurons by taking the input from previous layers. A final linear layer is added of which output neurons are 3 according to our need. After fine-tuning the model, we get the model parameters as below:

Total parameters 141,878,951

Total Non-Trainable parameters 139,570,240 Total Trainable parameters 2,308,711

Now we can visualize the graphical representation of the model that represents how a input data is handled by the model to give the final output.

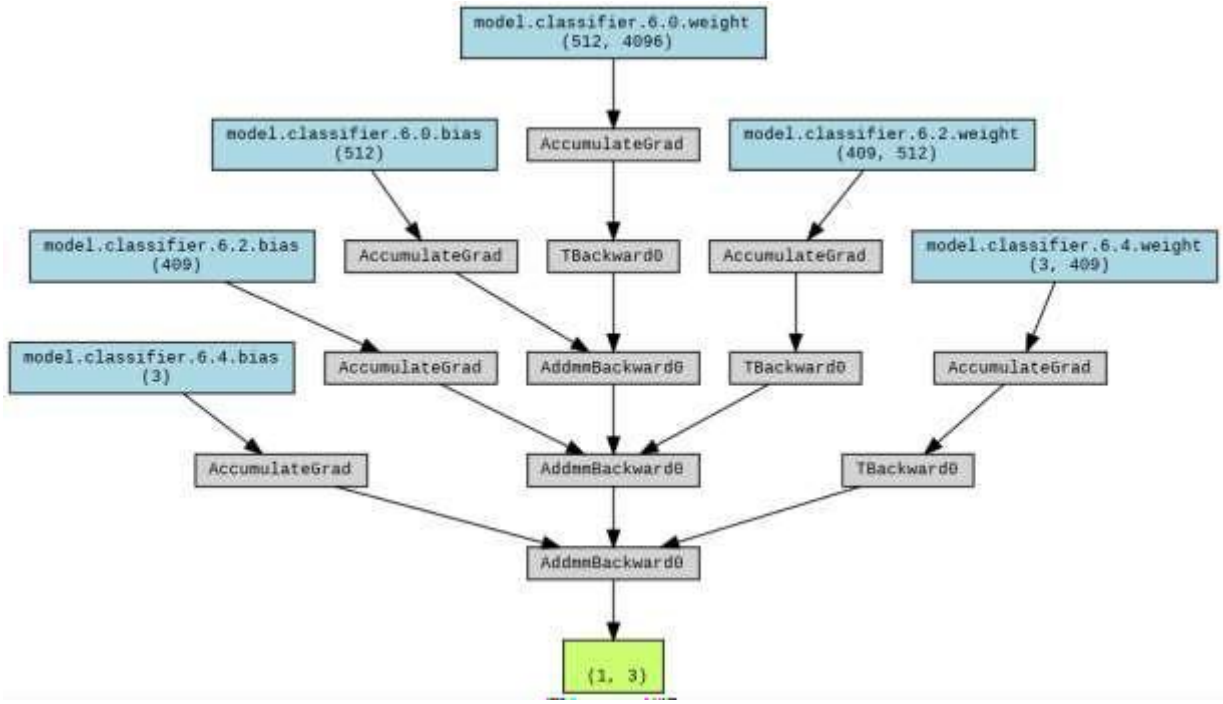


Figure 5.7 VGG19 model flow diagram

5.5 Model Training:

We sliced the datasets into a total of 110 batches of size 8 for training the model, total of 14 batches of size 8 for validation and total of 14 batches of size 8 for testing purposes. There were total of four models that were trained multiple times by tuning various hyperparameters in order to bring the best possible outcome from them. The first model is a custom CNN model, the second is DenseNet121 model, third is ResNet152 model and the fourth is VGG19 model. Some of the best possible common hyperparameter set to train this all models are epoch of 50, learning rate of 0.01, filter size of 3X3, batch size of 8 and Adam as optimizer.

Now after deciding and building the model architecture and proper hyperparameters tuning, the model is trained on the training datasets along with proper model validation using validation dataset. During model training the model parameters will learns multiple times through updating model weights and biases. We continued this process until and unless model performances on the validation data reach a satisfactory level or no further improvement was observed.

5.6 Software Development Model:

Prototype model supplies a working model to the user early in the process, enabling early assessment and increasing user's confidence. The developer gains experience and insight by developing a prototype there resulting in better implementation of requirements. The prototyping model serves to clarify requirements, which are not clear, hence reducing ambiguity and improving communication between the developers and users. There is a great involvement of users in software development. Hence, the requirements of the users are met to the greatest extent. It also helps in reducing risks associated with the software. We used this model while building the web application part of our project in order to create a user-friendly web interface.

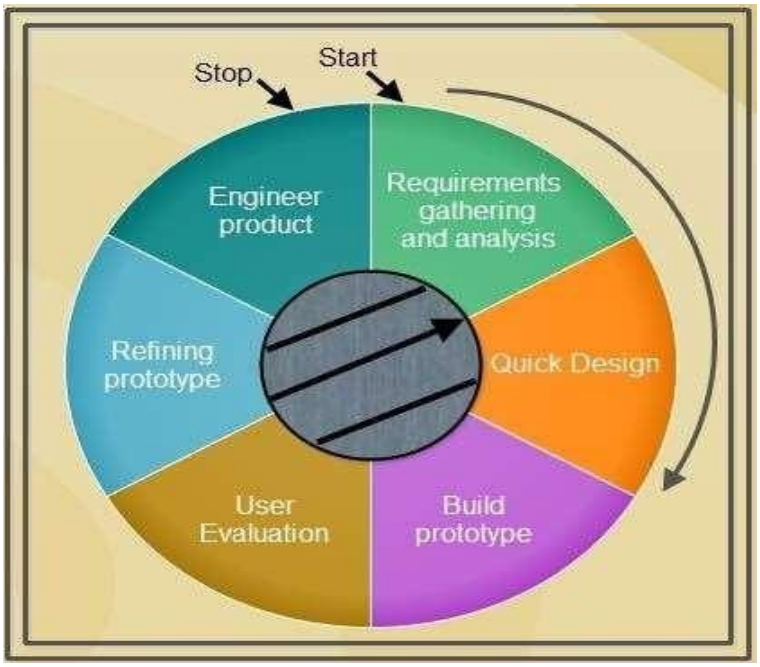


Figure 5.8 Prototype Model of Development

CHAPTER 6
IMPLEMENTATION

6.1 Environment Setup

To develop the lung cancer prediction system, the following Python libraries were used:

- **TensorFlow/Keras**: For building and training the CNN model.
- **OpenCV**: For reading and preprocessing CT scan images.
- **NumPy**: For numerical operations.
- **Matplotlib**: For visualizing model performance.
- **Scikit-learn**: For splitting the dataset.

```
import os, numpy as np, cv2, matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
```

6.2 Data Loading and Preprocessing

The dataset is organized into two categories: `Cancer` and `Non-Cancer`, representing CT images of cancerous and healthy lungs, respectively. Each image was converted to grayscale, resized to a standard resolution (128x128 pixels), and normalized to improve model training efficiency.

```
IMG_SIZE = 128
DATASET_PATH = "path_to_dataset"
CATEGORIES = ["Cancer", "Non-Cancer"]
```

```
def load_data():
    data, labels = [], []
    for category in CATEGORIES:
        path = os.path.join(DATASET_PATH, category)
        label = CATEGORIES.index(category)
        for img_name in os.listdir(path):
            try:
                img_path = os.path.join(path, img_name)
                img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
                img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
                data.append(img)
                labels.append(label)
```

```
                img_path = os.path.join(path, img_name)
                img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
                img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
                data.append(img)
                labels.append(label)
            except:
                continue
    return np.array(data), np.array(labels)

X, y = load_data()
X = X.reshape(-1, IMG_SIZE, IMG_SIZE, 1) / 255.0
```

6.3 Train-Test Splitting

To evaluate the model, the dataset was split into training and testing sets in an 80:20 ratio using `train_test_split`.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
```

6.4 CNN Model Architecture

The model was implemented using Keras' Sequential API. It consists of multiple convolutional and pooling layers for feature extraction, followed by fully connected dense layers for classification. Dropout is used to prevent overfitting.

```
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(IMG_SIZE, IMG_SIZE, 1)),
    MaxPooling2D(pool_size=(2,2)),

    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(pool_size=(2,2)),

    Conv2D(128, (3,3), activation='relu'),
    MaxPooling2D(pool_size=(2,2)),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

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

6.5 Model Training

The model was trained for 10 epochs with a batch size of 32. The validation data was used to monitor overfitting and performance generalization.

```
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=
```

6.6 Evaluation and Visualization

After training, the model was evaluated on the test set. Additionally, plots of accuracy and loss during training were generated for performance analysis.

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy * 100:.2f}%")

# Visualization
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title('Model Accuracy')
```

```
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Model Loss')

plt.show()
```

6.7 Prediction on New CT Images

The trained model can be used to classify new CT images as cancerous or non-cancerous. Below is a function that handles this prediction process

```
def predict_image(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
    img = img.reshape(1, IMG_SIZE, IMG_SIZE, 1) / 255.0
    prediction = model.predict(img)
    return "Cancer" if prediction[0][0] > 0.5 else "Non-Cancer"

# Example usage
print(predict_image("path_to_new_image.jpg"))
```

CHAPTER7
RESULTS AND DISCUSSION

7.1 Model Evaluation

Any machine/deep learning model results should be evaluated mathematically and statistically to authenticate the nature and usability of the model. The main aim behind model evaluation is to determine the performance of the model in accurately detecting lung cancer and to identify the areas for the further improvement of the model. Overall, this all helps to test the generalizing capacity of the developed model for various upcoming new unseen data points. The model evaluation also provides a deep insight on model strengths and weaknesses. So we tested various factors of the model in order to observe and evaluate the model closely.

7.1.1 Performance evaluation through training and validation curves

In a machine learning model, the training accuracy and loss are measures of how well the model is performing on the training data. Training accuracy is the fraction of correct predictions made by the model on the training data, while the training loss is a measure of the error made by the model on the training data.

The validation accuracy and loss are similar measures but are calculated on a separate dataset called the validation set. The validation set is used to evaluate the model's performance on unseen data, and is used to tune the model's hyper parameters (e.g. learning rate, number of hidden units, etc.).

It is important to monitor both the training accuracy/loss and the validation accuracy/loss during the training process. If the training accuracy is high but the validation accuracy is low, it could indicate that the model is overfitting to the training data (i.e. it is performing well on the training data but is not generalizing well to new data).

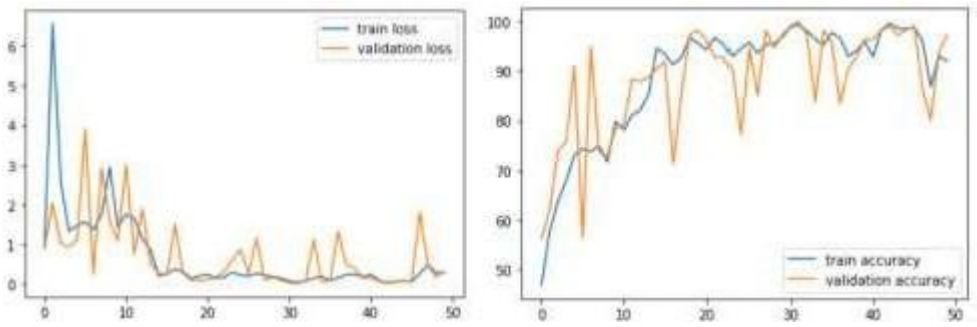


Figure 7.1 Training and Validation curves for custom model

Here, by looking at the loss curves the loss is constantly decreasing and in the accuracy curve accuracy is increasing with the train and validation alignment. It shows that the model is performing well.

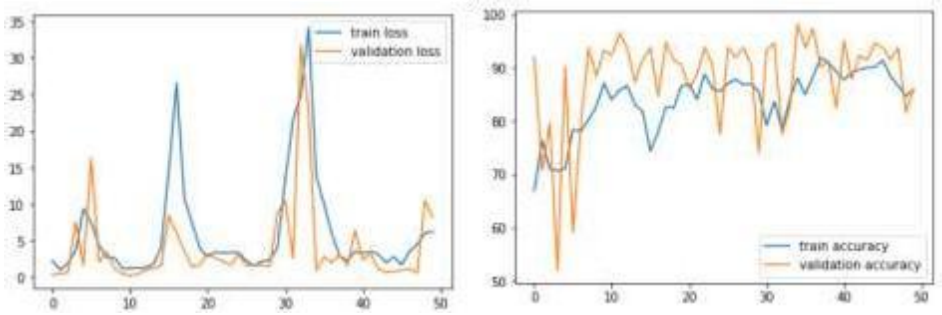


Figure 7.2 Training and Validation curves for DenNet121 tuned model

The model loss is fluctuating with large values at different epochs and its accuracy is increasing with both train and validation are properly aligned. It shows the model is doing good overall.

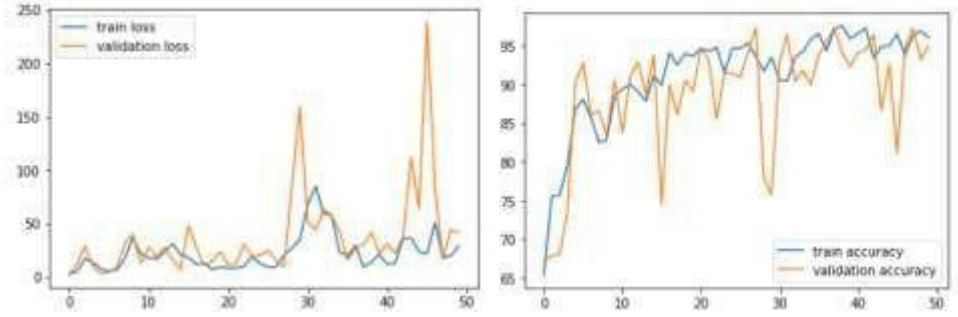


Figure 7.3 Training and Validation curves for ResNet152 tuned model.

Model loss is increasing and rapidly fluctuating after epoch 30 whereas its accuracy is constantly increasing. Here both the train and validation curves are properly aligned with each other though due to rapid fluctuations on loss with increments is undesirable condition.

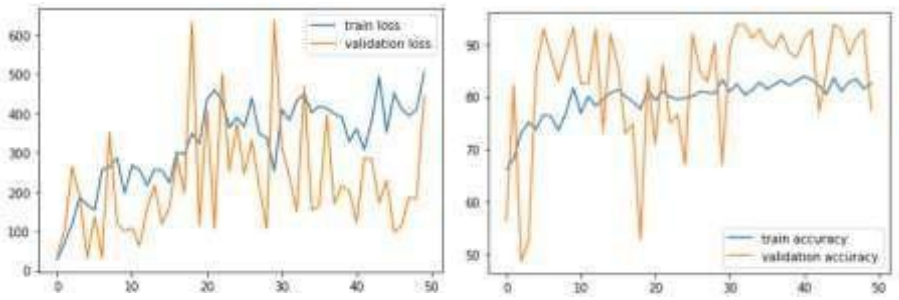


Figure 7.4 Training and Validation curves for VGG19 tuned model.

Here both model loss curves and accuracy curves show bad results and the performance of this model is worse and undesirable.

Overall, we can visualize that VGG19 tuned model (Figure 14) is suffering a lot and validation and training losses and accuracy are not aligning to each other. Also, we can observe that losses are increasing that makes this model worst of all. Figure 13 shows average performance and Figure 12, and Figure 11 indicate the good performance among all.

7.1.2 Model accuracy evaluation:

Model accuracy is the metric defined as the number of classifications a model correctly predicts divided by the total number of predictions made. It is one of the simplest and most used methods to evaluate the model. The model accuracy of four model trained are shown in following table:

model name	model loss	model acc
lung cancer custom model	0.54	92.86
Cancer DenseNetModel	3.91	89.15
Cancer ResNetModel	8.94	97.32
Cancer VGG19Model	333.68	79.46

Table 7.1 Various model and their accuracy

7.1.3 Confusion Matrix:

The confusion matrix of all the four models is presented below for the analysis of how well the model became able to successfully classify the testing data to their respective classes. The confusion matrix of the four models is listed below:

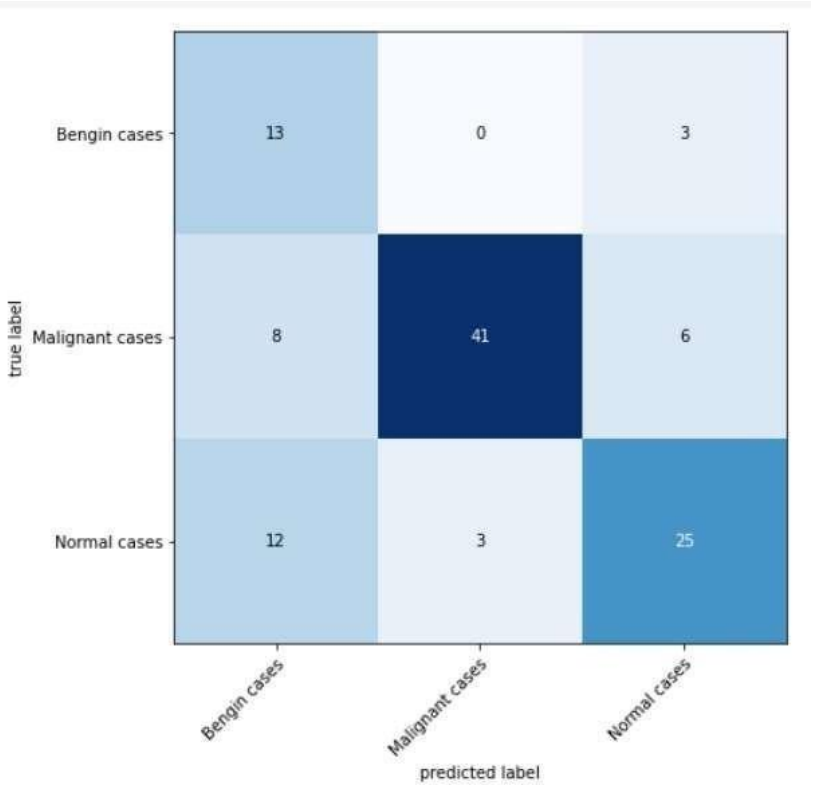


Figure 7.5 Confusion matrix of custom CNN model

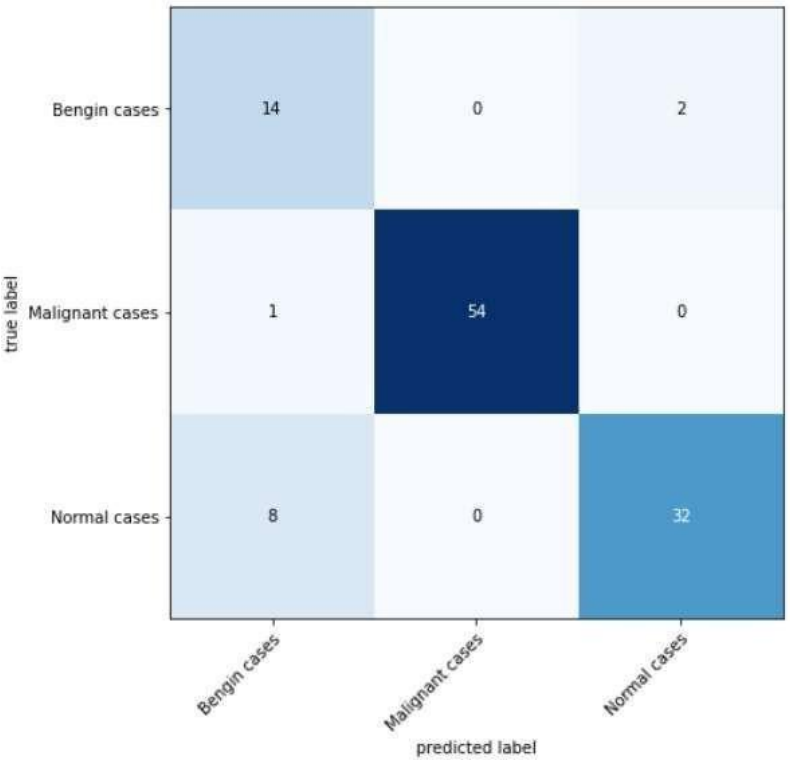


Figure 7.6: Confusion matrix of DenseNet121 tuned model

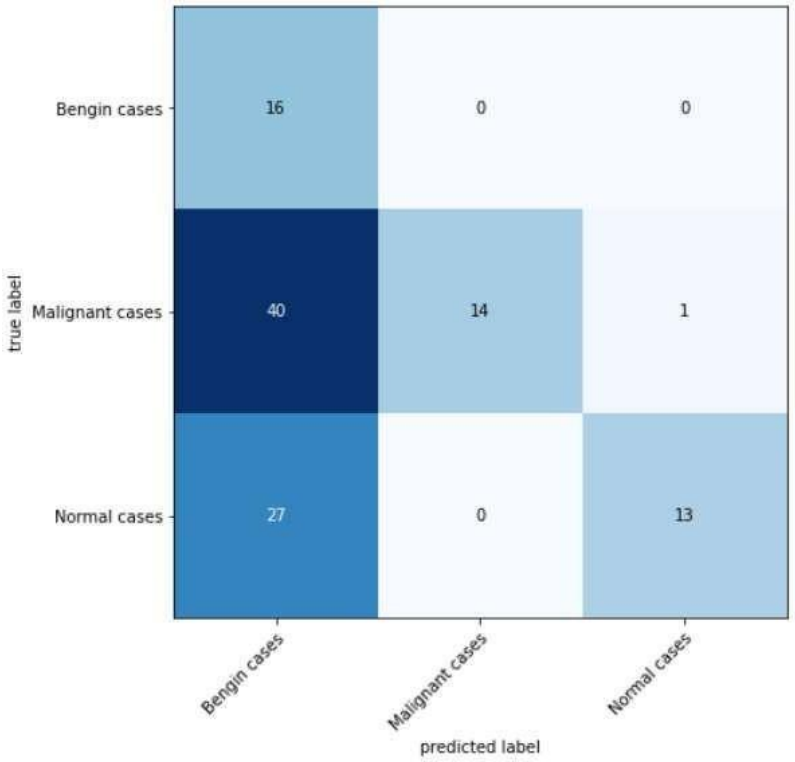


Figure 7.7 Confusion matrix of ResNet152 tuned model

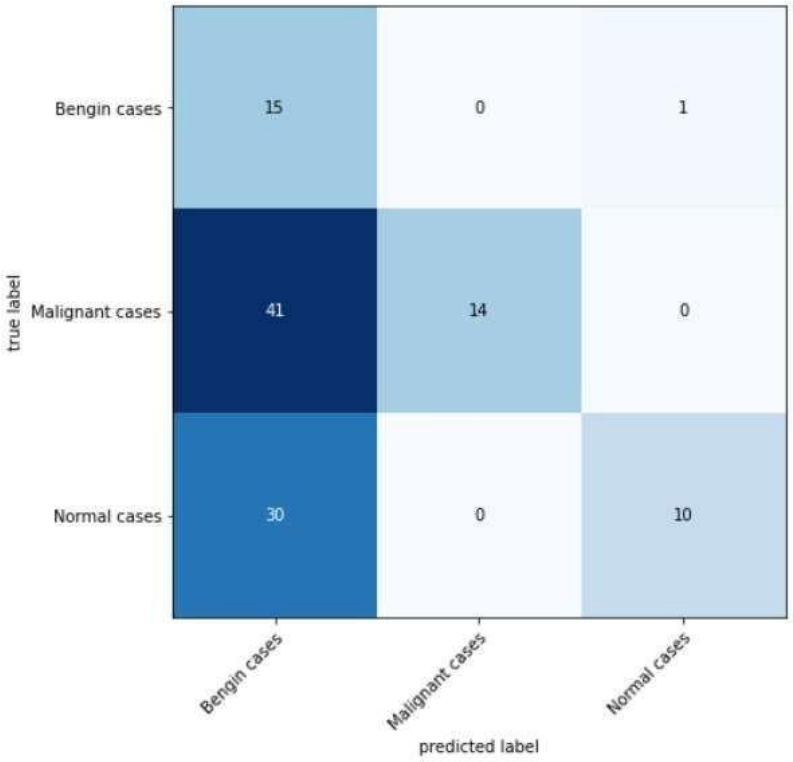


Figure 7.8 Confusion matrix of GGG19 tuned model

After visualizing the confusion matrix of the model on test dataset, we can observe that the best performing model is DenseNet121 tuned model. The worst performing models are VGG19 tuned model and ResNet152 tuned model. Although ResNet152 model has higher accuracy during training and validation process it did not perform well for the testing/unseen data. The custom CNN model works average as comparison to another trained model.

7.1.4 Precision, recall and f1 score of the model:

Since we are tackling the lung cancer prediction model, only accuracy is not enough metric to evaluate the model. It is equally essential for us to take care of other metrics such as precision recall and f1 scores. Among them, recall is one of the important parameters as it deals with the false negative (FN) value minimization that is, less the FN value greater the recall value and model will not predict non cancer CT scan as Cancerous CT scan.

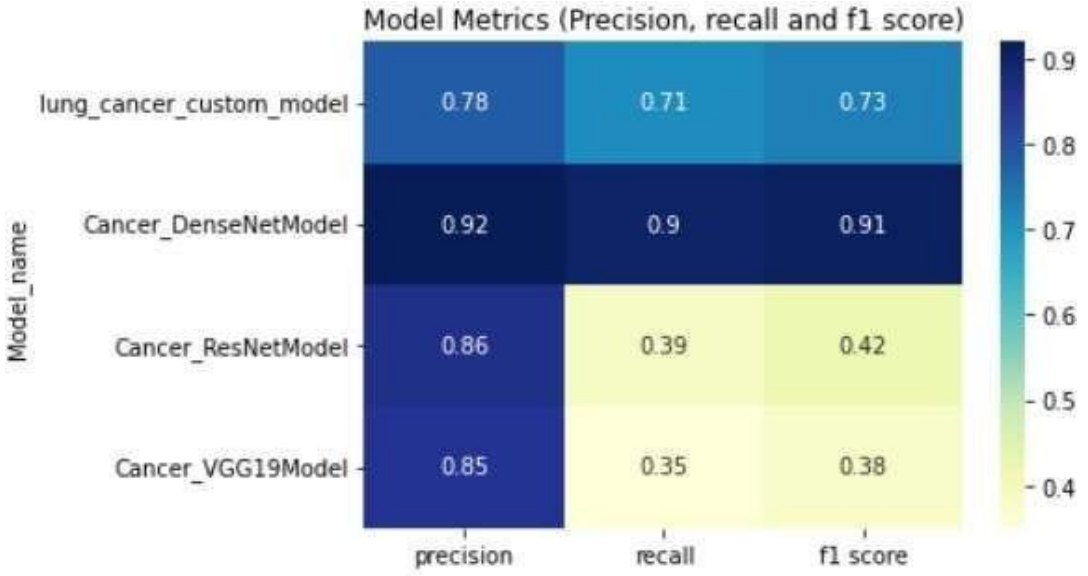


Table 7.2 Various Model Metrics (Precision, Recall and f1-score)

From the table we can visualize that all models have good precision but most of the model suffers in recall and f1 score parameter. The model that performs well in all precision, recall and f1 score is Cancer_DenseNetModel. The models that perform very badly in recall and f1 score is Cancer_VGG19Model and Cancer_ResNetModel. The Lung_cancer_custom_model performs average as compared to other models according the the statistics present in above figure.

7.1.5 ROC-AUC Curve:

The receiver operating characteristic (ROC) curve is a way to represent the relationship between sensitivity and specificity graphically. We evaluate the area under the ROC curve (AUC) to measure the performance of the model. The curves of the four models are presented below:

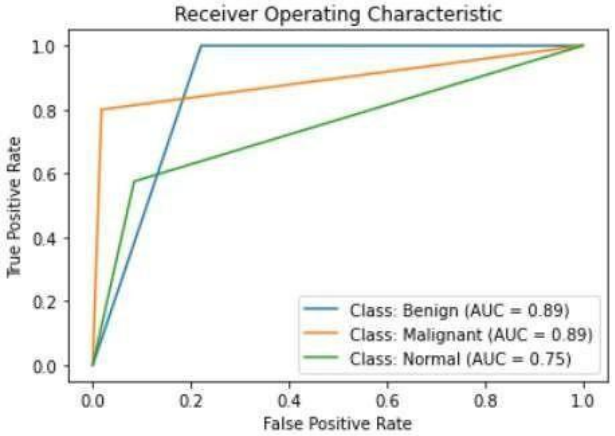


Figure 7.9 ROC-AUC curve for custom-cnn model

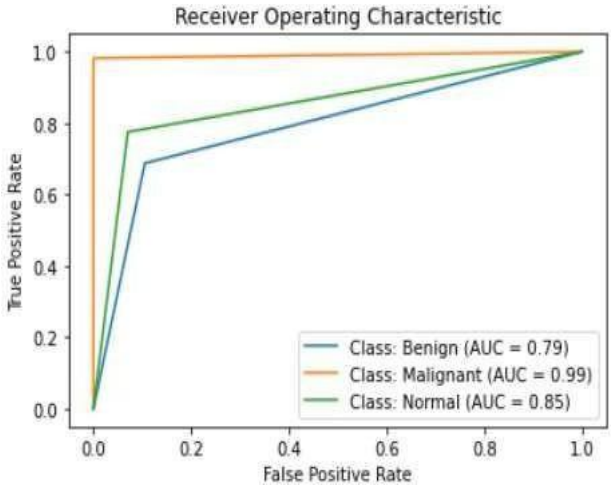


Figure 7.10 ROC-AUC curve for DenseNet121 tuned model

Here, among the two figure 20 shows quite the better results than figure 19. In the case of figure 19, benign case and malignant case got highest AUC score of 89% whereas normal case got the lowest of 75%. Here it shows that overall model performs descent to classify benign case and malignant case. In the case of figure 20 we got AUC of malignant case as 99% that means model performs very well and accurate enough to classify malignant case. The last case normal got AUC of 85% that means model is also performing well to classify normal cases.

7.1.6 Model Explanation Using Explainable AI:

Since machine learning and deep learning models are not explainable on itself, as a result it is very essential to use various explanation techniques in order to observe the model performance [25]. Explain ability is especially essential for our model as it is being used in the medical sector where precision and recall of the model prediction should be very high. It helps our model for informed clinical decision making and builds trust and transparency among the medical community. It also improves the model accuracy and reliability that also addresses the ethical concerns related to potential biases in healthcare system.

For XAI purpose we used SHAP (Shapley Additive exPlanations) . SHAP is a unified framework for interpreting machine learning models. It provides an explainable method for attributing a prediction to the features that influenced it. The framework is based on cooperative game theory and the concept of Shapley values, which provides a way to fairly distribute the prediction among the input features. SHAP values represent the contribution of each feature to the final prediction of the model, and these contributions can be positive as well as negative. Here if the contribution is positive then it increases the prediction probability else if it is negative than it decreases the prediction probability.

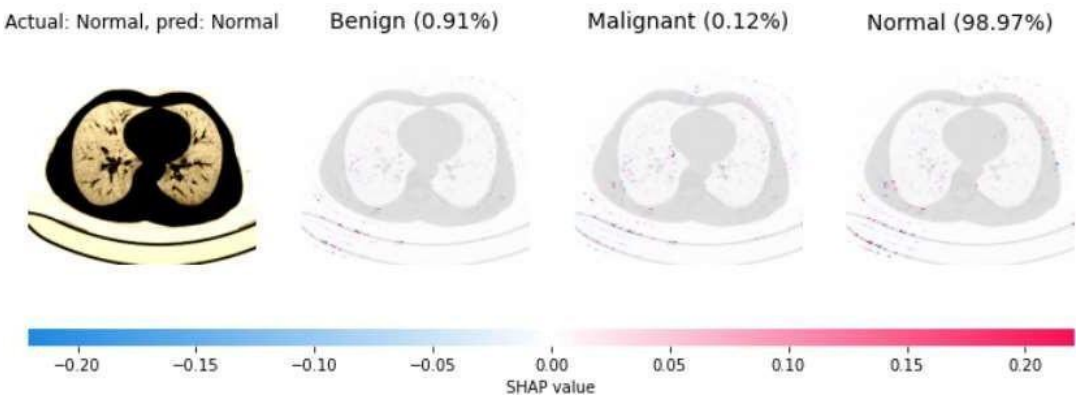


Figure 7.11 SHAP result for Normal case

Here, blue dots indicate negative features that reduce the probability of the predicted results being normal whereas the pink dots indicates positive features that increase the probability of the predicted results being normal. We can observe that in figure 19, pink dots are maximum in fourth image (98.97%) and the probability of being Normal is high. In second there are some pink dots but maximum there is blue dots.

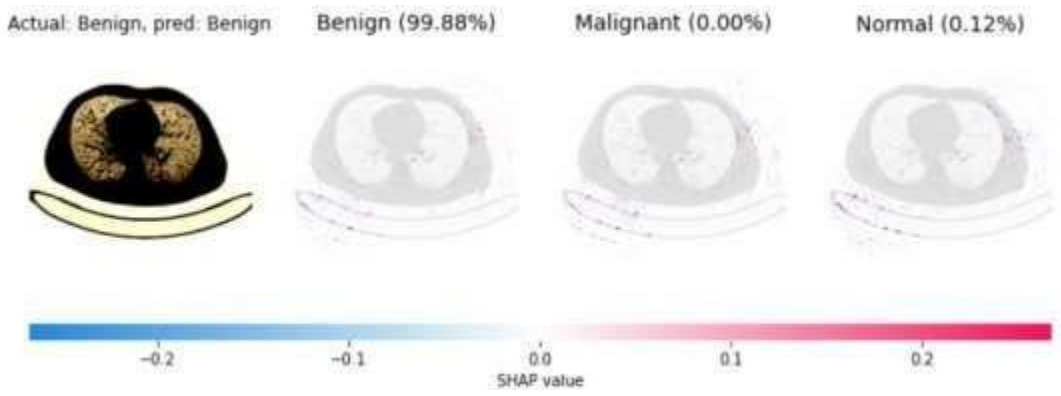


Figure 7.12 SHAP result for benign case

Here, the pink dots in second image are more with less blue dots making match with benign case with around 99.88%. The malignant case contains blue and pink dots in balancing amount and finally the normal case has some visible dots with around only 0.12% matching.

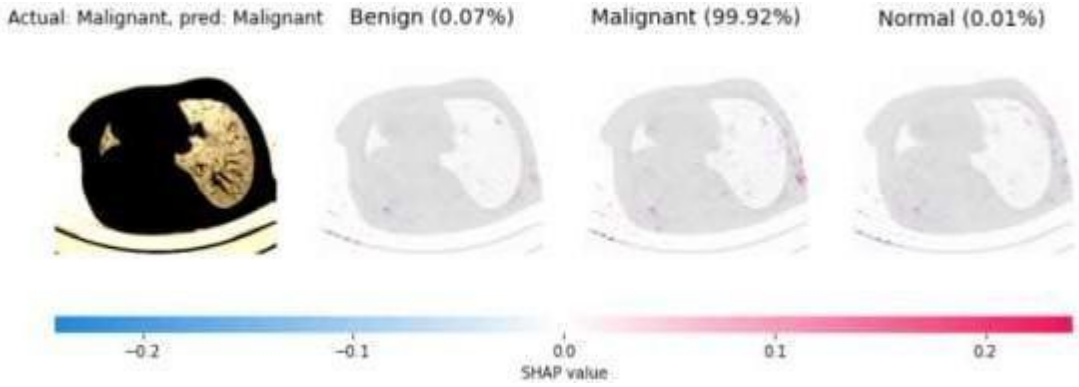


Figure 7.13 SHAP result for benign case

Here, the pink dots are dense in the third image showing 99.92% matching the property of malignant case of the given CT scan image. The Benign case contains some pink dots but most of them are blue dots and similarly in the case of Normal there are pink dots in some part but most of them are blue dots.

7.2 Development of web user interface:

The web application is created using streamlit. Here users can learn about lung cancer, test their chest CT scans image for cancer. There is also an added section to visualize the model and its performance. It is developed with the

motive to provide an easy visual interface to the user to insert the CT scan image and test it.

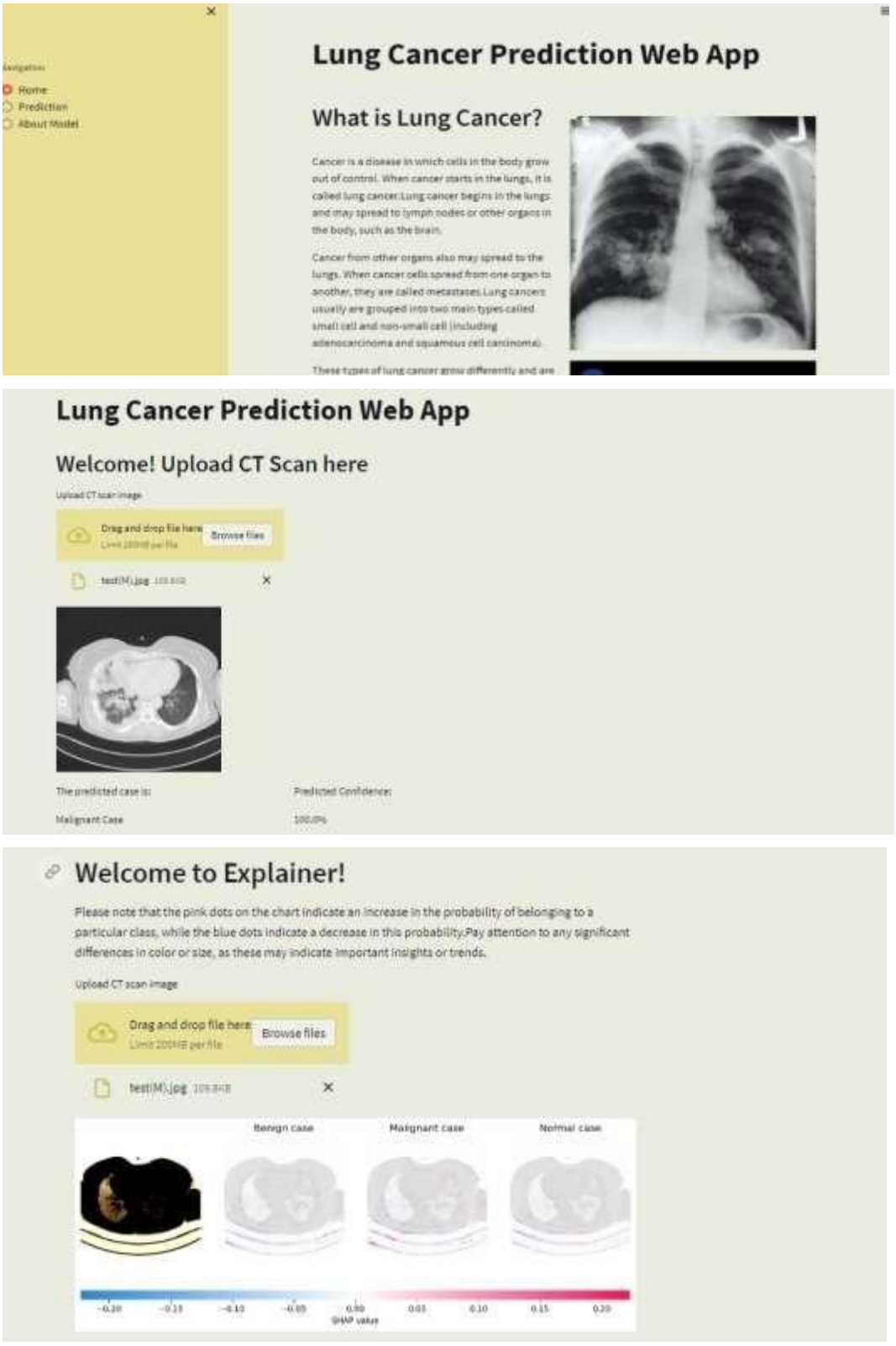


Figure 7.14: Web application interface screenshot

7.3 Discussion:

A faster approach with AI to help in the initial screening of the chest CT scans to detect the lung cancer can help the radiologist to speed up their daily screening task as well as they can supervised the screening task of more places such as rural and underserved areas of Nepal. This project whole idea was to investigate on the potential approach to fulfill this criterion in order to make the more effective healthcare system in Nepal.

In contrast to conventional procedures like X-rays and CT scans for detecting lung cancer manually, the CNN model offers a more economical and streamlined alternative. Its implementation could allow healthcare providers to precisely detect malignant pulmonary nodules in less time. The various studies that we have gone through during our project we found that most of them have less accuracy or use other supporting machine learning algorithms by extracting features from the CT scans, as our main aim was to study useability of CNN in detecting lung cancer and our approach is purely based on it. Since we realized that only accuracy is not sufficient to evaluate the model performance, and have evaluated various other parameters including precision, recall, f1 score, ROC_AUC, and overall confusion matrix.

In addition, we have implemented the concept of explainable AI using SHAP. This will assist medical personnel in visualizing why the model is providing specific output and which parameters of the CT scan are contributing to the result. The incorporation of explainable AI will improve the model's output and enhance interpretability for medical personnel, allowing them to manually visualize the infected area for further analysis and confirmation. Moreover, medical practitioners can employ the CNN model as an auxiliary resource to augment their screened case count. This approach has the possibility of advancing healthcare systems in remote and neglected regions where established screening procedures may either be absent or ascertainable costs are prohibitive.

CHAPTER 8
CONCLUSIONS

The developed model is trained on available dataset and various model evaluating performance parameters such as accuracy, precision, recall, F1 score and ROC-AUC curve were studied. From Custom_Model we got accuracy of 92.86%, precision of 78%, recall of 71%, and f1 score of 73%. From Cancer_DenseNetModel we got accuracy of 89.15%, precision of 92%, recall of 90%, and f1 score of 91%. From Cancer_ResNetModel we got accuracy of 97.32%, precision of 86%, recall of 39%, and f1 score of 42%. Finally, from Cancer_VGG19Model we got accuracy of 79.46%, precision of 85%, recall of 35%, and f1 score of 38%. From the listed figures we can conclude that Cancer_DenseNetModel and Custom_Model performed well. We further evaluated the ROC-AUC curve for this two best performing models and got the average of 0.84 for Custom_model and average of 0.87 for Cancer_DenseNetModel. This high value of precision and recall, ROC-AUC analysis values indicate CNN model can effectively capture image parameters and detect lung cancer. This has huge potential for early detection and improved patient outcomes on implementing in the medical field. It will be a helpful tool for medical personnel and improve healthcare in rural and underserved areas.

CHAPTER 9

LIMITATION AND FUTURE ENHANCEMENTS

9.1 Limitation

In our project while deep driving into it, we found several limitations that include a lot of further research to improve the performance of the model that makes it more robust and can easily generalize. Since we are dealing with healthcare systems, which is an extremely sensitive topic and should be implemented only after a lot of tests and validation. Some major limitation includes:

Data availability and quality: The quality and quantity of the data used for training and testing the model can greatly impact its performance.

Model robustness and generalization: Despite the high accuracy of the model, it may not perform well on new, unseen data, indicating a lack of robustness and generalization.

Time and Resources: Training a model with a large amount of dataset takes a lot of time. We are required to have advanced resources like GPU and TPU, but they are either too expensive or not always available.

9.2 Future Enhancements:

In future we can enhance this model by fitting more variety and tons of datasets, which can make the model more relevant and powerful to predict unseen data. We can further implement more complex combined algorithms such as feature extraction and use with machine learning algorithm for the measurement of the present tumor size and nature of it, through

CHAPTER 10

APPENDIX

10.1 Source code

```
!apt-get install unrar

!unrar x "/content/drive/MyDrive/lung_image_sets1.rar" "/content/extract/"

import os
import cv2
import numpy as np

# Define correct dataset path
DATASET_PATH = "/content/extract/lung_image_sets1"
IMG_SIZE = (224, 224)

def load_and_preprocess_images(directory):
    images, labels = [], []

    for class_name in os.listdir(directory):
        class_path = os.path.join(directory, class_name)

        if os.path.isdir(class_path):  # Ensure it's a folder
            for img_name in os.listdir(class_path):
                img_path = os.path.join(class_path, img_name)

                # Debugging: Check if image exists
                if not os.path.exists(img_path):
                    print(f"File not found: {img_path}")
                    continue

                # Read and preprocess image
                img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
                if img is None:
                    print(f"Error loading: {img_path}")
                    continue

                img = cv2.resize(img, IMG_SIZE)
                img = img / 255.0  # Normalize

                images.append(img)
                labels.append(class_name)

    return np.array(images), np.array(labels)
```

```
X, y = load_and_preprocess_images(DATASET_PATH)
```

```
# Reshape for CNN
X = X.reshape(-1, IMG_SIZE[0], IMG_SIZE[1], 1)

print(f"Loaded {X.shape[0]} images, each of shape {X.shape[1:]}")

from sklearn.model_selection import train_test_split

# Split into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)

print(f"Training data: {X_train.shape[0]} images")
print(f"Testing data: {X_test.shape[0]} images")

from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder

# Convert labels to numbers
label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_test = label_encoder.transform(y_test)

# Convert to one-hot encoding
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

print("Labels converted to categorical format!")

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout

# Define CNN model
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(224, 224, 1)),
    MaxPooling2D(2,2),

    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),

    Conv2D(128, (3,3), activation='relu'),
    MaxPooling2D(2,2),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),  # Prevent overfitting
```

```
    Dense(len(set(y_train.argmax(axis=1))), activation='softmax')  # Multi-
class classification
])

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

# Model Summary
model.summary()

# Train the CNN model
history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=8,
    batch_size=32
)
# Evaluate the model on test data
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {test_acc*100:.2f}%")

# Save the model as an H5 file
model.save("/content/lung_cancer_cnn.h5")

print("Model saved successfully!")

mport cv2
import numpy as np

def predict_image(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (224, 224))  # Resize to match CNN input size
    img = img / 255.0  # Normalize
    img = img.reshape(1, 224, 224, 1)  # Reshape for model input

    # Make prediction
    prediction = model.predict(img)
    class_index = np.argmax(prediction)  # Get predicted class
    confidence = np.max(prediction) * 100  # Get confidence score
(percentage)
    class_label = label_encoder.inverse_transform([class_index])[0]

    # Determine Cancer or No Cancer
    if "lung_n" in class_label.lower():  # Assuming "lung_n" means normal (no
cancer)
        status = "No Cancer"
    else:
```

```
        status = "Cancer Detected"

    print(f"Predicted Class: {class_label} ({confidence:.2f}% confidence) -
{status}")

# Test with a sample image
predict_image("/content/extract/lung_image_sets1/lung_n/lungn1.jpeg"

pip install gradio
import gradio as gr
import cv2
import numpy as np
from tensorflow.keras.models import load_model

# Load your trained model
model = load_model("/content/lung_cancer_cnn.h5")

# Define class labels manually (ensure this matches your training order)
class_names = ["lung_n", "lung_scc", "lung_aca"]  # Modify if needed

def predict_image_gradio(image):
    # Convert to grayscale
    image = image.convert("L")
    img = np.array(image)
    img = cv2.resize(img, (224, 224))
    img = img / 255.0
    img = img.reshape(1, 224, 224, 1)

    # Predict
    prediction = model.predict(img)
    class_index = np.argmax(prediction)
    confidence = np.max(prediction) * 100
    class_label = label_encoder.inverse_transform([class_index])[0]

    # Determine status
    if "lung_n" in class_label.lower():
        status = "🟢 No Cancer"
    else:
        status = "🔴 Cancer Detected"

    return f"Class: {class_label}\nConfidence: {confidence:.2f}%\nStatus:
{status}"

# Gradio interface
interface = gr.Interface(
    fn=predict_image_gradio,
    inputs=gr.Image(type="pil"),
    outputs="text",
```

10.2 screenshot

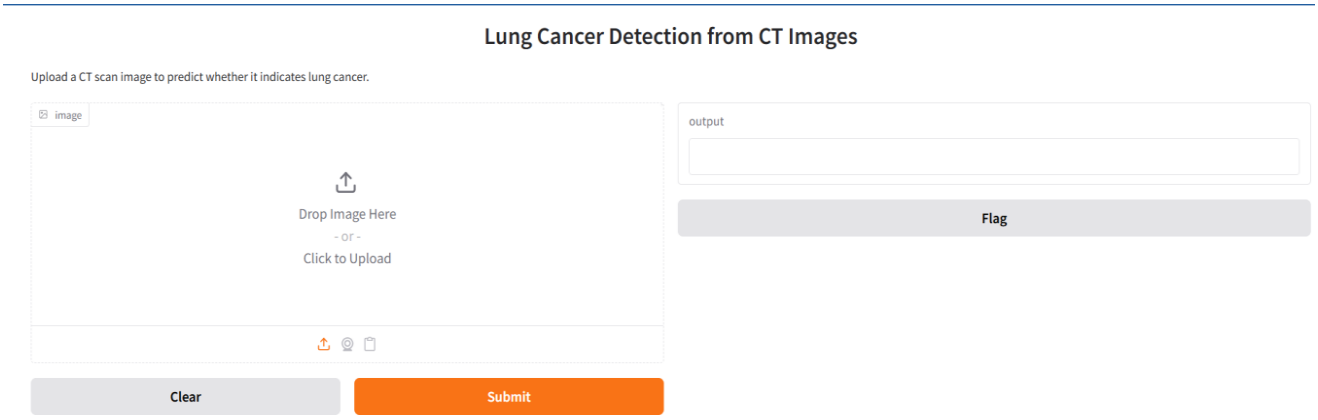


Figure 10.1 Upload Image Page

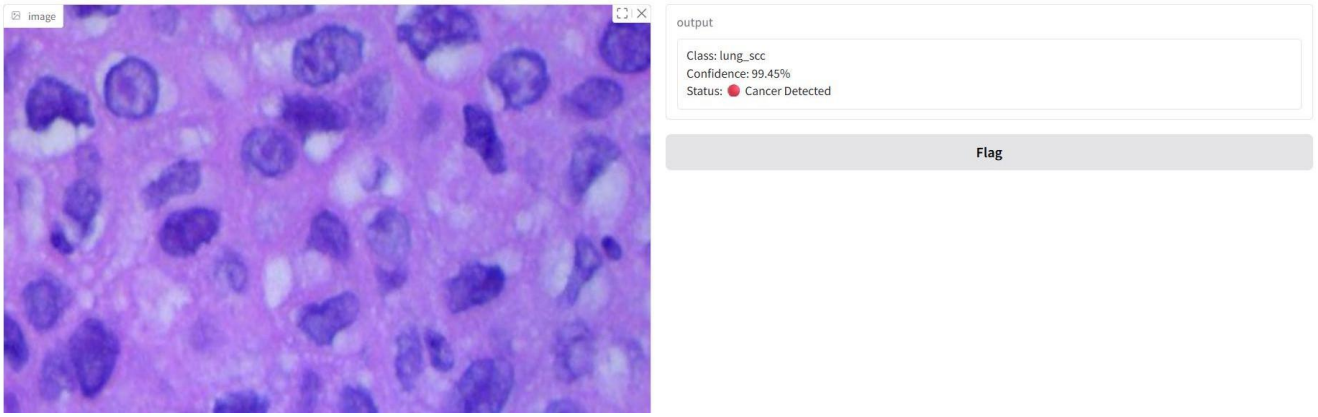


Figure 10.2: SCC Prediction Image

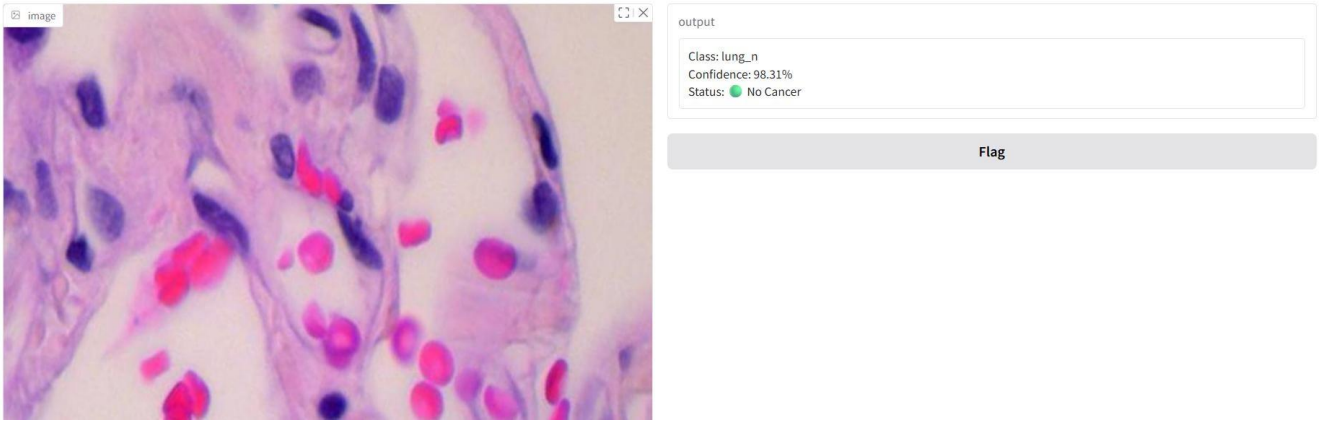


Figure 10.3 Normal Prediction Image

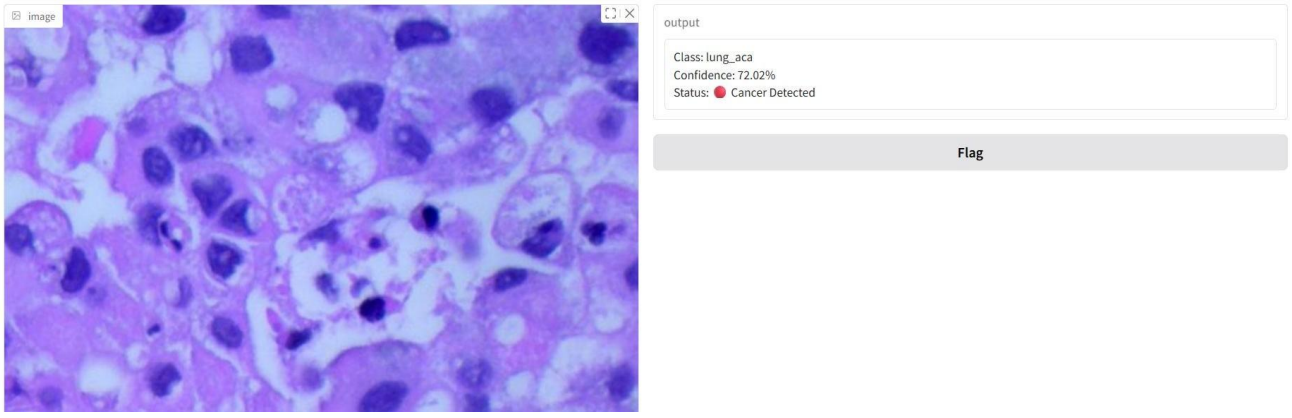


Figure 10.4 ACA Prediction Image

CHAPTER 11
REFERENCES

[1] WHO,"WorldHealthOragnization,"[Online].Available:https://www.who.int/news-room/fact-sheets/detail/cancer. [Accessed 11 06 2022].

[2] "Verywellhealth," [Online]. Available: https://www.verywellhealth.com/benign lung-tumors-4691959#:~:text=Characteristics%20and%20Behavior&text=Size%3A%20While %20malignant%20tumors%20are,time%20of%20roughly%204%20months.. [Accessed 11 06 2022].

[3] N. B. o. Disease, 2019. [Online]. Available: https://nhrc.gov.np/wp content/uploads/2022/02/BoD-Report-Book-includ-Cover-mail- 6_compressed.pdf.

[4] Globocan,2020.[Online].Available:https://gco.iarc.fr/today/data/factsheets/populat ions/524-nepal-fact-sheets.pdf.

[5] D. S. W. Joseph A. Cruz, "Applications of Machine Learning in Cancer Prediction and Prognosis," *Cancer Informatics*, no. 2, p. 59–78, 2006.

[6] D. C. e. al., "Multi-column Deep Neural Networks for Image Classification," 2012.

[7] A. Geron, Hands-on machine learning with Scikit-Learn, Keras and Tensorflow 2019.

[8] X. W. e. al, "An Appraisal of Lung Nodules Automatic Classification Algorithms for CT images," 2019.

[9] N. M. ,. D. S. S. D. D. N Kalaivani1*, "Deep Learning Based Lung Cancer Detection and," in *Materials Science and Engineering*, 2020.

[10] M. S. A.-H. F. Y. M. E. A. K. Z. S. H. Hamdalla F. Al-Yasriy, "Diagnosis of Lung Cancer Based on CT Scans Using CNN," in *2nd International Scientific Conference of Al-Ayen University (ISCAU-2020)*, 2020.

[11] D. B. 2. M. C. Y.-C. B. Eali Stephen Neal Joshua, "3D CNN with Visual Insights for Early Detection of Lung Cancer Using Gradient-Weighted Class Activation," *Journal of Healthcare Engineering*, vol. 2021, no. 6695518, p. 11, 2021.

[12] D. S. e. al., "Lung Cancer Detection Using Convolutional Neural Networ," *INTERNATIONAL RESEARCH JOURNAL OF ENGINEERING AND TECHNOLOGY (IRJET)* , vol. 9, no. 01, 2022.

[13] "wikipedia,"2022.[Online].Available:https://en.wikipedia.org/wiki/Artificial_intelligence. [Accessed 10 December 2022].

[14] Wikipedia,"Wikipedia,"2022.[Online].Available:https://en.wikipedia.org/wiki/Deep_learning. [Accessed 22 December 2022].

[15] Wikipidia,"wikipidia,"2022.[Online].Available:https://en.wikipedia.org/wiki/Convolutional_neural_network. [Accessed 22 December 2022].

[16] V. a. G. Hinton, ""Rectified linear units improve restricted boltzmann machines"," in *27th International Conference on Machine Learning (ICML-10)*, 2010.

[17] L. B. Y. B. a. P. H. Y. LeCun, ""Gradient-based learning applied to documen recognition,"," *Journal of Multivariate Analysis,* vol. 86, no. 11, 1998.

[18] B. Adhikari, "Prediction of Potential Heart Disease using Ensemble Learning," Kathmandu, 2021.

[19] 2022. [Online]. Available: https://data.mendeley.com/datasets/bhmdr45bh2/1 [Accessed 11 March 2022].

[20] Y. J. O. V. J. H. N. Z. E. T. a. T. D. J. Donahue, "Decaf: A deep convolutiona activation feature for generic visual recognition," in *the International Conference on Machine Learning (ICML)*, 2014.

[21] Z. L. K. Q. W. a. L. v. d. M. G. Huang, "Densely Connected Convolutiona Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] X. Z. S. R. a. J. S. Kaiming He, "Deep Residual Learning for Image Recognition," in *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[23] K. S. a. A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *the International Conference on Learning Representations (ICLR)*, 2015.

[24] [Online]. Available: https://ecomputernotes.com/software-engineering/explain- prototyping-model.

[25] C. Garbin, "Assessing Methods and Tools to Improve Reporting, Increase Transparency, and Reduce Failures in Machine Learning Applications in Healthcare," 2020.

[26] S. M. &. L. Lundberg, " A unified approach to interpreting model predictions," *Advances in neural information processing systems*, 2017.

CT Lung Cancer Prediction using CNN

Mr.Saravana Kumar .M
BE CSE (Final year)
mrsarvan87@gmail.com
Dr.G.U. Pope College of Engineering

Mr.Arockia Jerin .J
BE CSE (Final year)
arockiyajerin26@gamil.com
Dr.G.U. Pope College of Engineering

Mrs.Sheebha.S.N
Assistant Professor/CSE
Dr.G.U. Pope College of Engineering

***Abstract*- Lung cancer remains a leading cause of global morbidity and mortality, necessitating advanced diagnostic tools for early detection and intervention. This project introduces a novel approach to lung cancer prediction by leveraging Convolutional Neural Networks (CNN) on lung X-ray images. The CNN model is trained to classify lung X-rays into four categories: adenocarcinoma, large cell carcinoma, normal, and squamous cell carcinoma. The methodology involves preprocessing of lung X-ray images, feature extraction using CNN layers, and classification through a carefully designed neural network architecture. The model's training is based on a dataset encompassing a diverse range of cases, facilitating robust learning and accurate predictions. The significance of this project lies in its potential to enhance early detection and streamline diagnosis, leading to timely medical interventions. By automating the categorization of lung cancer types through CNN analysis of chest X-rays, this system aims to contribute to the efficiency of healthcare services and improve patient outcomes. The evaluation of the proposed model involves comprehensive performance metrics, including accuracy, sensitivity, specificity, and precision. Results from extensive testing on diverse datasets showcase the model's efficacy in distinguishing between various lung cancer types and identifying normal cases.**

Keywords -CNN, Neural network, lung cancer, deep learning algorithm

I. INTRODUCTION

Lung sickness is a significant general wellbeing concern around the world, with pneumonia, cellular breakdown in the lungs, and persistent obstructive pneumonic illness (COPD) affecting huge number of individuals every year. Early and right conclusion of these problems is basic for effective treatment and the board. Profound learning, a kind of man-made consciousness (simulated intelligence), has arisen as a powerful method for clinical picture handling, with the possibility to increment indicative exactness and proficiency. Convolutional Neural Network (CNN) are a kind of profound learning model that is especially valuable for picture handling. CNNs might learn confounded examples and characteristics related with different lung illnesses by using enormous information bases of clinical pictures.

A growing number of people are interested in using CNNs to predict lung diseases, which could revolutionize medical diagnosis and treatment planning. CNNs expect the arrangement of profound learning-based lung. The's undertaking will probably make a model that can precisely foresee the presence or nonappearance of lung sickness utilizing clinical imaging information like lung X-Rays or CT scan. Via preparing the CNN on a fluctuated and delegate assortment of lung examines, we desire to use the capability of profound figuring out how to improve demonstrative exactness and convey early treatments to patients with lung issues.

Eventually, the objective is to make a trustworthy and productive instrument that will help medical care professionals successfully analyze and oversee lung issues, at last working on understanding results and personal satisfaction.

II. LITERATURE REVIEW

"Segmentation and Prediction of Lung Cancer CT Scans through Nodules using Ensemble Deep Learning Approach",the paper is The focus on division and prediction of cellular disintegration in the lungs. CT examination of knobs using a group profound learning approach yields promising results. We achieved greater precision and power in knob division and disease forecasting by coordinating many models. This method has the potential to improve early detection and analysis of cellular breakdown in the lungs, providing great assistance to radiologists and doctors. Further examination and approval are justified to assess its clinical suitability across diverse patient populations and medical care settings.

"A Deep Learning Based Approach to Predict Lung Cancer from Histopathological Images". The results of a study on the application of deep learning to the prediction of lung cancer from histopathological pictures would probably indicate that deep learning models are capable of reliably identifying and predicting the presence of lung cancer through effective analysis of these images. By providing an accurate and timely diagnosis, these models have the potential to improve patient outcomes by enabling earlier intervention and treatment

"A Lightweight Deep Learning Model for Automatic Diagnosis of Lung Cancer". The project focuses on developing a lightweight deep learning model for the automatic diagnosis of lung cancer. Extensive research and review have shown that the model is effective and efficient at accurately categorising

lung cancer cases based on medical imaging data. The proposed approach appears to be a promising method for speeding up and improving the accuracy of lung cancer diagnosis. This could result in early detection and better outcomes for patients. Further testing and approval in clinical settings are required to fully assess the therapeutic utility and versatility of the suggested paradigm.

III. EXISTING SYSTEM

LUNA16: The Lung Nodule Analysis (LUNA16) challenge was a widely recognized competition in the field of lung cancer detection from CT scans. While it's not a system per se, it provided a benchmark dataset and a platform for researchers to develop and evaluate lung cancer prediction algorithms. Many research papers and code implementations have been built upon this challenge.

DeepLung: DeepLung is an open-source project developed by researchers at the University of Central Florida, aimed at detecting lung nodules from CT scans using deep learning techniques. It provides a TensorFlow-based implementation of various deep learning models for lung nodule detection and classification.

CheXNet: Although primarily focused on chest X-ray analysis for various pathologies, including pneumonia, CheXNet, developed by researchers at Stanford University, demonstrated the potential of deep learning in medical image analysis. While not specifically for lung cancer prediction, the techniques and methodologies used in CheXNet could be adapted for similar tasks.

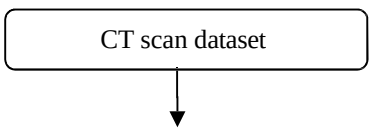
IBM Watson for Oncology: IBM Watson for Oncology is a commercial system that utilizes artificial intelligence, including deep learning algorithms, to assist oncologists in making treatment decisions for cancer patients. While it's not solely focused on lung cancer prediction, it showcases the integration of AI into clinical decision-making processes.

Google AI's Research on Lung Cancer Prediction: Google's DeepMind Health team has conducted research on using deep learning algorithms to predict lung cancer from CT scans. While their work is primarily research-oriented, it highlights the potential of deep learning in improving early detection and diagnosis of lung cancer.

IV. PROPOSED SYSTEM

The cellular breakdown in the lungs forecast framework comprises of a few primary parts, every one of which is expected to work on the prescient model's general exactness and utility. The main module includes information preprocessing, in which chest X-Ray pictures are exposed.

PROPOSED BLOCK DIAGRAM:



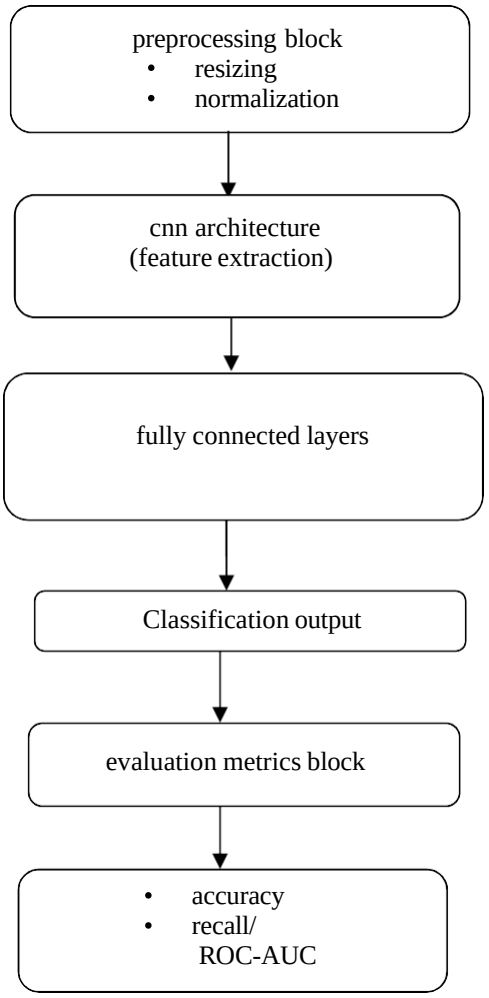


Fig. 1. Block Diagram of Proposed System

Thorough cleaning, standardization, and expansion cycles to guarantee the dataset's optimal quality and consistency. This stage is basic for working on the model's ability to identify minor examples that demonstrate unmistakable types of cellular breakdown in the lungs. It focuses on the design and preparation of the Convolutional Neural Network (CNN). This includes the careful selection and design of lungs network layers, such as convolutional layers for highlight extraction and fully related layers for characterization. The model is based on a distinct dataset that includes adenocarcinoma, large cell carcinoma, squamous cell carcinoma, and typical instances. optimisation techniques are employed during training to fine-tune the model's parameters for best performance.

After the preparation stage, the third module centers around coordinating and conveying the prepared CNN model. The development of an interface that enables healthcare providers to submit chest X-ray images for real-time analysis is the objective of this module. The program creates expectations for each picture, sorting it as both of the predefined cellular breakdown in the lungs sorts

incorporate easy to understand points of interaction, versatility, and consistent correspondence between the model and the client

The fourth module involves rigorous testing and validation. The presentation of the framework is evaluated using many metrics such as exactness, awareness, particularity, and accuracy. Testing covers a variety of datasets to ensure the model's power and generalizability. Criticism from clinical professionals is also sought to further enhance the model and address any potential issues in certifiable clinical situations.

At last, the fifth module covers the understanding and representation of results. This involves getting pertinent experiences from the model's expectations, for example, recognizing huge areas in X-Ray pictures that add to order. Representation methods are utilized to help clinical professionals appreciate and trust the model's decisions, expanding straightforwardness and interpretability.

This technique ensures an ordered and comprehensive improvement process, which includes information planning, model creation, organisation, testing, and result translation. The combination of these modules aims to express a refined and solid cellular breakdown in the lungs' expectation framework, with the potential to significantly influence early conclusion and work on comprehending findings.

V. SOFTWARE USED

Anaconda, as mentioned earlier, is not a specific tool for cancer prediction or deep learning. However, it provides an environment conducive to developing such projects. If you're working on cancer prediction using deep learning, particularly Convolutional Neural Networks (CNNs), Anaconda can be extremely helpful in managing your Python environment and dependencies.

Environment Setup: Anaconda allows you to create isolated Python environments, which is beneficial for managing dependencies. You can create a new environment specifically for your cancer prediction project, ensuring that you have all the necessary libraries installed without conflicting with other projects or system-wide packages.

Installing Deep Learning Libraries: Anaconda's package manager, conda, and its associated package repository, conda-forge, provide easy access to popular deep learning libraries like TensorFlow, PyTorch, and Keras. You can use conda to install these libraries into your project environment.

Data Handling and Preprocessing: Utilize libraries like pandas for data handling and manipulation. For image data, you might use libraries like OpenCV to load and preprocess images before feeding them into your CNN model.

Model Development: With deep learning libraries like TensorFlow or PyTorch installed via Anaconda, you can develop CNN models for cancer prediction. These libraries provide high- level APIs for building neural networks, including convolutional layers for image analysis.

Training and Evaluation: Train your CNN model on your dataset using the capabilities of TensorFlow or PyTorch. Anaconda's environment management ensures that you have all the required dependencies for training and evaluation in a consistent environment.

Visualization and Analysis: Anaconda includes libraries like matplotlib and seaborn, which are useful for visualizing data and model performance metrics. You can use these libraries to analyze the results of your cancer prediction model.

Overall, while Anaconda itself isn't specifically designed for cancer prediction or deep learning, it provides a powerful environment for managing dependencies and developing deep learning models using libraries like TensorFlow or PyTorch, which are commonly employed in such projects.

VI. PROTOTYPE MODEL

INPUT:

Fig 2: Input image 1

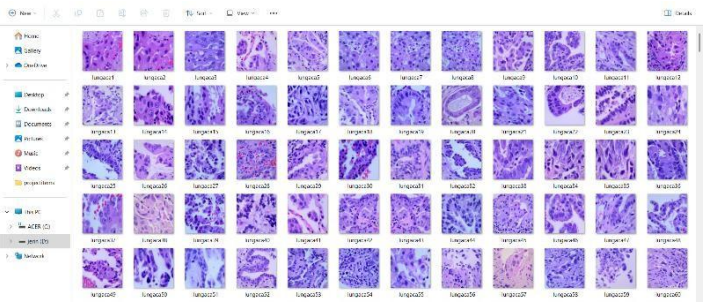


Fig 3: Input image 2

OUTPUT

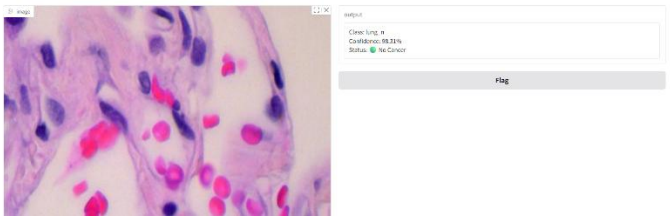


Fig 4 : Output image

VII Results And Discussions

In result of, this project represents a significant stride in the realm of medical diagnostics, specifically in the early detection of lung cancer using X-rays. By harnessing the power of Convolutional Neural Networks (CNN), our system demonstrates a robust capacity to classify diverse lung cancer types—adenocarcinoma, large cell carcinoma, squamous cell carcinoma—and normal cases with notable accuracy.

The meticulously designed modules, ranging from data preprocessing to model deployment and result interpretation, collectively form a comprehensive pipeline. Through extensive testing on varied datasets and continual refinement based on medical professional feedback, the system exhibits not only promising predictive capabilities but also a potential for practical integration into clinical workflows.

The fusion of advanced technology with healthcare in this project underscores the transformative impact that artificial intelligence can have on early diagnosis, thereby offering a valuable tool to the medical community in the ongoing battle against lung cancer. This project stands as a testament to the convergence of cutting-edge technology and healthcare, with the overarching goal of improving patient outcomes and contributing to the advancement of medical science

VIII Conclusion

In conclusion, this work offers a tremendous progression in the field of clinical diagnostics, remarkably the early determination of cellular breakdown in the lungs using X-Ray. It utilizes Convolutional Neural Network (CNN) to characterize different cellular breakdown in the lungs types — adenocarcinoma, enormous cell carcinoma, and squamous cell carcinoma — as well as expected cases with high precision. The exactly evolved modules, which range from information preprocessing to demonstrate sending and result understanding, make up an exhaustive pipeline. Through far reaching testing on different datasets and progressing change in light of clinical expert criticism, the framework shows great prescient abilities as well as the potential for commonsense mix into clinical work processes. The undertaking's combination of refined innovation and medical services features the emotional impact that man-made brainpower can have on early conclusion, giving a significant device to the clinical local area in the continuous conflict against cellular breakdown in the lungs. This drive epitomizes the crossing point of state of the art innovation and medical services, with a definitive objective of working on persistent results and adding to the progression of clinical science.

REFERENCES

[1] Muthazhagan B, Ravi T, Rajinigirinath D. An enhanced computer-assisted lung cancer detection method using content-based image retrieval and data mining techniques. Journal of Ambient Intelligence and Humanized Computing. June 2 2020 :pp 1-9.

[2] Masud M, Sikder N, Nahid AA, Bairagi AK, AlZain MA. A machine learning approach to diagnosing lung and colon cancer using a deep learning-based classification framework. Sensors. Jan;21 2021 .

[3] Sajja T, Devarapalli R, Kalluri H. Lung Cancer Detection Based on CT Scan Images by Using Deep Transfer Learning. Traitement du Signal. Oct;36(4):339-44 2019

[4] Sajja T, Devarapalli R, Kalluri H. Lung Cancer Detection Based on CT Scan Images by Using Deep Transfer Learning. Traitement du Signal. Oct;36(4):339-44 2019

[5] Detection and classification of lung cancer using CNN and Google netR. Pandian *, V. Vedanarayanan, D.N.S. Ravi Kumar, R. Rajakumar

[6] Deep learning ensemble 2D CNN approach towards the detection of lung cancer AsgharAli Shah1, HafzAbid MahmoodMalik2*,AbdulHafeez

[7] Muhammad1AbdullahAlourani3 & Zaeem Arif Butt1 CT Scan Images by Using Deep Transfer Learning. Traitement du Signal. Oct;36(4):339-44 2019

[8] Ahmed Medjahed S., AitSaadi T., Benyettou A., Ouali M. Kernel-based learning and feature selection analysis for cancer diagnosis. *Applied Soft Computing* . 2017;51:39doi: 10.1016/j.asoc.2016.12.010

JAYARAJ ANNAPACKIAM CSI COLLEGE OF ENGINEERING

(Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai)

MARGOSCHIS NAGAR, NAZARETH-628 617.

NCISCT - 2025

CERTIFICATE OF APPRECIATION

This is to certify that

Mr. /~~Ms.~~ /~~Mrs.~~ M. SARAVANA KUMAR

of Dr. G.U. Pope College of Engineering Participated / Presented

a paper entitled CT Lung Cancer Prediction using CNN. in the

NATIONAL CONFERENCE ON INTELLIGENT SYSTEMS AND COMMUNICATION TECHNOLOGIES (NCISCT) 2025, organized by the Department of Electronics and Communication Engineering on 6th May 2025.

Co-ordinators	Head of the Department	Principal
Dr. M. Ruban Gladwin Mrs. R. Jeya Preeda	Dr. K. Agnes Prema Mary	Dr. S. Jeyakumar

JAYARAJ ANNAPACKIAM C.S.I. COLLEGE OF ENGINEERING • NAZARETH 628 617

Designed by: B. Onisavathi, III ECE B
T. Selva Kumari, III ECE B

CERTIFICATE

JAYARAJ ANNAPACKIAM CSI COLLEGE OF ENGINEERING

(Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai)

MARGOSCHIS NAGAR, NAZARETH-628 617.

NCISCT - 2025

CERTIFICATE OF APPRECIATION

This is to certify that

~~Mr.~~ /Ms. /~~Mrs.~~ J.AROCKIA JERIN

of Dr.G.U. Pope College of Engineering Participated / Presented

a paper entitled CT Lung Cancer Prediction using CNN in the

NATIONAL CONFERENCE ON INTELLIGENT SYSTEMS AND COMMUNICATION TECHNOLOGIES (NCISCT) 2025, organized by the Department of Electronics and Communication Engineering on 6th May 2025.

Co-ordinators
Dr. M. Ruban Gladwin
Mrs. R. Jeya Preeda

Head of the Department
Dr. K. Agnes Prema Mary

Principal
Dr. S. Jeyakumar

JAYARAJ ANNAPACKIAM C.S.I. COLLEGE OF ENGINEERING NAZARETH 628 617

Designed by: B. Onisavathi, III ECE B
T. Selva Kumari, III ECE B