

WAVELET- BASED LUNG CANCER CLASSIFIER WITH GESTURE INTERFACE



A DESIGN PROJECT REPORT

Submitted by

SANJANA M

SWATHI S

SWETHA S

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

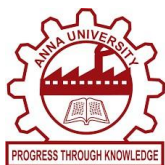
COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, Affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER, 2024



WAVELET- BASED LUNG CANCER CLASSIFIER WITH GESTURE INTERFACE



A DESIGN PROJECT REPORT

Submitted by

SANJANA M (811721104088)

SWATHI S (811721104110)

SWETHA S (811721104112)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, Affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER, 2024

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled “**WAVELET - BASED LUNG CANCER CLASSIFIER WITH GESTURE INTERFACE**” is the bonafide work of **SANJANA M (811721104088), SWATHI S (811721104110), SWETHA S (811721104112)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report 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. A. Delphin Carolina Rani, M.E., Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K. Ramakrishnan College of Technology

(Autonomous)

Samayapuram – 621 112.

SIGNATURE

Ms.P. Karthika, M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K. Ramakrishnan College of Technology

(Autonomous)

Samayapuram – 621 112.

Submitted for the viva-voce examination held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We jointly declare that the project report on **“WAVELET- BASED LUNG CANCER CLASSIFIER WITH GESTURE INTERFACE”** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **“ANNA UNIVERSITY CHENNAI”** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the award of Degree of **BACHELOR OF ENGINEERING**.

Signature

SANJANA M

SWATHI S

SWETHA S

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Technology (Autonomous)**”, for providing us with the opportunity to do this project.

We are glad to credit honorable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our project and offering adequate duration in completing our project.

We would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

We whole heartily thanks to **Dr. A DELPHIN CAROLINA RANI, M.E., Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

We express my deep and sincere gratitude to our project guide **Ms.P. KARTHIKA, M.E.**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for her incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course. We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

An automated lung cancer diagnosis system utilizing CT scan images is an advanced diagnostic technology designed to identify lung cancer and determine its severity. By applying cutting-edge image processing and machine learning techniques, the system accurately analyzes CT scan images to classify cancer stages and provide diagnostic insights. The integration of a gesture-based interface enhances usability, allowing intuitive and hands-free interaction with the system. This innovation is particularly valuable in healthcare, supporting medical professionals in early detection and effective management of lung cancer. Leveraging advancements in computer vision, artificial intelligence, and human-computer interaction, this system has the potential to revolutionize cancer diagnosis, improve diagnostic accuracy, and contribute to better clinical outcomes.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	ix
	LIST OF ABBREVIATIONS	x
1	INTRODUCTION	1
	1.1 BACKGROUND	1
	1.2 OVERVIEW	2
	1.3 PROBLEM STATEMENT	2
	1.3 OBJECTIVE	3
	1.4 IMPLICATION	3
2	LITERATURE SURVEY	4
	2.1 MULTI – ORIENTATION LOCAL TEXTURE FEATURES FOR GUIDED ATTENTATION BASED FUSION IN LUNG NODULE CLASSIFICATION	4
	2.2 LUNG CANCER DETECTION USING MACHINE LEARNING	4
	2.3 AN ENHANCED COMPUTER ASSISTED LUNG CANCER DETECTION USING CONTENT BASED IMAGE RETRIVAL TECHNIQUES	5
	2.4 A COMPARATIVE STUDY OF LUNG CANCER DETECTION USING MACHINE LEARNING	5
	2.5 STUDY ON IN – VITRO LUNG CANCER DETECTION USING E-NOSE TECHNOLOGY	6
	2.6 SELF – UPGRADED CAT MOUSE OPTIMIZER WITH MACHINE LEARNING DRIVEN LUNG CANCER	6

3	SYSTEM ANALYSIS	7
	3.1 EXISTING SYSTEM	7
	3.1.1 PROBLEMS ON EXISTING SYSTEM	8
	3.2 PROPOSED SYSTEM	9
	3.3 BLOCK DIAGRAM FOR PROPOSED SYSTEM	9
	3.4 APPROACH USED	10
	3.5 PROCESS CYCLE	10
	3.6 FLOWCHART	11
	3.7 USE CASE DIAGRAM	12
4	MODULES	13
	4.1 MODULE DESCRIPTION	13
	4.1.1 PREPROCESSING	13
	4.1.2 IMAGE SEGMENTATION	14
	4.1.3 FEATURE SELECTION	15
	4.1.4 CLASSIFICATION	15
	4.1.5 GESTURE CONTROL	16
	4.1.6 DECISION - MAKING	17
	4.2 CNN ALGORITHM	18
	4.2.1 FEATURE EXTRACTION PHASE	19
	4.2.1 CLASSIFICATION PHASE	19
	4.3 HAAR CASCADE ALGORITHM	19
5	SYSTEM SPECIFICATION	21
	5.1 SOFTWARE REQUIREMENTS	21
	5.1.1 GOOGLE COLAB	21
	5.1.2 GOOGLE DRIVE	22
	5.1.3 OPEN CV	23
	5.1.4 MEDIA PIPE	24
	5.1.5 PyAutoGUI	25
	5.2 HARDWARE REQUIREMENTS	26
6	METHODOLOGY	27
	6.1 DATA COLLECTION AND DATASET PREPARATION	27

	6.2 PREPROCESSING OF CT SCAN IMAGES	27
	6.3 FEATURE EXTRACTION	28
	6.4 FEATURE SELECTION	29
	6.5 CLASSIFICATION USING DEEP LEARNING	29
	6.6 GESTURE CONTROL INEGRATION	30
	6.7 POST- PROCESSING AND DECISION MAKING	31
	6.8 SYSTEM INTEGRATION AND TESTING	31
	6.9 EVALUATION AND OPTIMIZATION	32
7	CONCLUSION AND FUTURE ENHANCEMENT	33
	7.1 CONCLUSION	33
	7.2 FUTURE ENHANCEMENT	34
	 APPENDICES I	 35
	APPENDICES II	43
	REFERENCES	48

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
3.1	BLOCK DIAGRAM	8
3.2	PROCESSING OVERVIEW	9
3.3	PROCESS AND RESULT	10
3.4	FUNCTIONS INVOLVED AND ACTIONS	11
4.1	FUNCTIONS OF PREPROCESSING	13
4.2	GESTURE CONTROL ACTION	16

LIST OF ABBREVIATIONS

CT	-	COMPUTED TOMOGRAPHY
CNN	-	CONVOLUTIONAL NEURAL NETWORK
SU-CMO	-	SELF- UPGRADED CAT MOUSE OPTIMIZER
MFO	-	MOUTH FLAME OPTIMIZATION
SSO	-	SLAP SWARM OPTIMIZATION
KNN	-	K- NEAREST NEIGHBOURS
DT	-	DECISION TREE
LR	-	LOGISTIC REGRESSION
NB	-	NAVIE BAYES
SVM	-	SUPPORT VECTOR MACHINE
AI		ARTIFICIAL INTELLIGENCE
ML		MACHINE LEARNING
E-nose		ELECTRONIC NOSE
HCI		HUMAN COMPUTER INTERACTION
ROI		REGION OF INTEREST
PPI		PIXEL PER INCH

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Lung cancer remains one of the leading causes of cancer-related deaths globally, making early detection and accurate diagnosis crucial for improving survival rates. Traditional methods of analyzing CT scans for lung cancer, which rely heavily on radiologists, are time-consuming and subject to human error. The advent of Convolutional Neural Networks (CNNs) has revolutionized medical image analysis, particularly in detecting lung cancer. CNNs are deep learning models that excel in extracting intricate features from medical images, enabling automatic and accurate identification of abnormalities such as tumors and nodules in CT scans. This technology significantly reduces the workload for radiologists, increases diagnostic speed, and improves accuracy. Furthermore, integrating a gesture-based interface allows users to interact with the system hands-free, enhancing accessibility and usability, especially in clinical settings. By combining the power of CNNs for accurate diagnosis with intuitive gesture control, this system offers a more efficient, reliable, and user-friendly approach to lung cancer detection and severity assessment.

Input CT scan Image	Image Preprocessing	Feature Extraction	Output Result	Gesture Controlled Interface
---------------------	---------------------	--------------------	---------------	------------------------------

1.2 OVERVIEW

Users who do not have a physical mouse can still interact with the system using a gesture-controlled interface, which utilizes a webcam to track eye movements. This hands-free method eliminates the need for traditional input devices such as a mouse or keyboard. The system uses Wavelet transforms (WT) for feature extraction from CT scan images, enabling the analysis and classification of lung cancer. By applying Convolutional Neural Networks (CNNs), the system detects cancerous abnormalities in the images and classifies their severity, assisting healthcare professionals in making timely decisions.

The gesture interface captures both static and dynamic eye gestures, allowing users to interact with the system without physical contact. The system can easily be controlled through hand gestures. This innovative approach to diagnosis combines wavelet-based image processing and deep learning (DL) models, ensuring accuracy, speed, and an efficient workflow for lung cancer detection.

1.3 PROBLEM STATEMENT

Lung cancer detection from CT scan images is critical yet prone to delays and errors due to manual interpretation. Existing tools often require physical input devices, unsuitable for sterile environments. This project addresses the need for an automated, accurate diagnostic system with wavelet-based image processing and a gesture-controlled interface, ensuring hands-free, efficient, and user-friendly cancer diagnosis.

1.4 OBJECTIVE

The primary objective of this project is to design a system that detects and interprets eye gestures for hands-free interaction and effective command execution in lung cancer diagnosis. This involves analyzing CT scan images using wavelet-based processing and CNNs for accurate classification. Additionally, the project emphasizes minimizing hardware costs by utilizing accessible tools like webcams, ensuring an efficient, user-friendly, and cost-effective solution for healthcare applications.

1.5 IMPLICATION

The implementation of this project involves multiple stages, beginning with data acquisition of CT scan images. These images undergo preprocessing, including noise removal and enhancement using wavelet-based techniques, to extract relevant features. The processed images are then input into a Convolutional Neural Network (CNN), which is trained to classify the presence and severity of lung cancer.

A gesture-controlled interface is developed in parallel, utilizing a standard webcam to capture eye gestures. The gestures are interpreted using image processing algorithms to enable hands-free interaction with the system. Static and dynamic gestures are mapped to commands such as navigation, zooming, and action selection.

The integration of CNN-based classification with the gesture interface ensures accurate results while maintaining ease of use. The implementation is tested rigorously to ensure reliability, efficiency, and adaptability in clinical environments.

CHAPTER 2

LITERATURE SURVEY

2.1 TITLE : Multi-Orientation Local Texture Features for Guided Attention-Based Fusion in Lung Nodule Classification

AUTHORS : Ahmed Saihood, Hossein Karshenas, Ahmad Reza Naghsh-Nilchi

YEAR : 2023

Computerized tomography (CT) scans are essential for lung cancer detection and classification. Lung nodules exhibit significant texture and size variations, challenging traditional methods. This study introduces a multi-orientation-based guided-attention module (MOGAM) and texture feature descriptors (TFDs) to model nodule dependencies across slices. Experiments on the LIDC-IDRI dataset show enhanced accuracy and AUC, offering explainable and effective results for medical experts.

2.2 TITLE : Lung Cancer Detection using Machine Learning

AUTHORS : Bharathy S, Pavithra R, Akshaya B

YEAR : 2022

Lung disease, particularly cancer, is a leading cause of death, and accurate diagnosis is challenging for radiologists. An intelligent computer-aided system supports radiologists in detecting lung cancer using machine learning (ML) techniques. The system uses multi-stage classification, segmentation methods like thresholding and marker-controlled watershed, and classifiers such as SVM, K-NN, and Random Forest, achieving an 88.5% accuracy with Random Forest.

2.3 TITLE : An Enhanced Computer- Assisted Lung Cancer Detection Using
Content-based Image Retrieval Techniques

AUTHORS : B. Muthazhagan, T. Ravi, D. Rajiniginath

YEAR : 2020

Cancer, especially lung cancer, is a leading global cause of death, and early detection remains a challenge. This paper proposes an efficient computer-assisted method using lung CT images, adaptive thresholding segmentation, and Support Vector Machine (SVM) for tumor classification. Additionally, content-based image retrieval is employed for feature comparison. A data mining approach further improves prediction accuracy, aiming to provide more effective and reliable detection of lung cancer for better clinical outcomes.

2.4 TITLE : A Comparative Study of Lung Cancer Detection using Machine
Learning Algorithms

AUTHORS : Radhika P.R, Rakhi A.S. Nair, Veena G.

YEAR : 2019

Lung cancer, caused by the uncontrolled growth of lung cells, has a high mortality rate, particularly among smokers. Early detection is vital for improving survival rates. This paper focuses on evaluating the performance of classification algorithms like Naive Bayes, SVM, Decision Tree, and Logistic Regression in predicting lung cancer. By analyzing the effectiveness of these algorithms, the aim is to enhance early diagnosis and improve patient outcomes, contributing to more accurate detection and timely intervention for lung cancer.

2.5 TITLE : Study on In-Vitro Lung Cancer Detection using E-nose technology

AUTHORS : R. Thriuman, A. Zakaria, A.I. Jeffree, N.A. Hishamuddin

YEAR : 2015

This study investigates the use of an electronic nose (E-nose) system to detect volatile organic compounds (VOCs) for early lung cancer diagnosis. The Cyranose 320 E-nose, equipped with 32 conducting polymer sensors, was used to identify and classify VOCs from lung cancer (A549) and breast cancer (MCF7) cell lines. By utilizing the k-Nearest Neighbors (KNN) classification algorithm, the E-nose successfully differentiated between VOC profiles with over 90% accuracy in just 30 seconds. The results suggest that the E-nose can rapidly and effectively detect VOCs associated with cancer cells, providing a promising tool for early diagnosis.

2.6 TITLE : Self-Upgraded Cat Mouse Optimizer with Machine Learning Driven Lung Cancer

AUTHORS : Mahmoud Ragab, Iyad Katib, Daa Hamed

YEAR : 2009

This project presents a lung cancer detection system utilizing the Self-Upgraded Cat Mouse Optimizer (SU-CMO) and machine learning for accurate CT scan classification. Wavelet-based preprocessing enhances feature extraction, while SU-CMO optimizes feature selection. A gesture-based interface enables hands-free operation, offering a cost-effective and adaptable solution. This system aims to improve diagnostic accuracy and streamline healthcare workflows, providing an efficient tool for early lung cancer detection and enhancing clinical decision-making processes.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

There are several existing systems and technologies for Lung cancer prediction, which are implemented with multiple algorithms,

1. K-Nearest Neighbors (KNN):

- Classifies lung cancer nodules by comparing features of CT scan images to their neighbors.
- Performance can degrade with noisy data and large datasets due to its instance-based nature.

2. Decision Tree (DT):

- Builds a flowchart model to classify lung cancer nodules based on features from CT scans.
- Simple and interpretable, but may overfit on small or imbalanced datasets, reducing accuracy.

3. Logistic Regression (LR)

- Used for binary classification, modeling the probability of malignancy in lung cancer.
- Assumes a linear relationship between features and outcomes, which may overlook complex patterns in CT data.

4. Naive Bayes (NB):

- A probabilistic classifier that calculates the likelihood of cancer based on Bayes' Theorem.
- Assumes feature independence, which may not hold true for CT scan data, leading to decreased accuracy.

3.1.1 PROBLEMS ON EXISTING SYSTEM

1. Limited Accuracy :

Existing systems often fail to provide consistently accurate results, especially when handling complex or ambiguous lung nodule features. Traditional classification algorithms like KNN, Decision Tree, and Naïve Bayes struggle with overlapping data points and feature interdependence.

2. High False Positives/Negatives:

The existing models, being less advanced in segmentation and classification techniques, are prone to higher false positive or negative rates, leading to potential misdiagnoses and delayed treatments.

3. Sensitivity to Noise:

Algorithms like KNN are highly sensitive to noise in the dataset, which can degrade their performance, especially with large or imbalanced datasets.

4. Overfitting Risk:

Decision Tree algorithms are prone to overfitting, particularly when trained on limited or imbalanced datasets. This reduces the generalization capability of the model for unseen data.

5. Linear Assumptions:

Logistic Regression assumes linear relationships between features and outcomes, which may not be suitable for capturing the complex patterns of lung cancer nodules in CT scans.

3.2 PROPOSED SYSTEM

The proposed system uses deep learning, specifically Convolutional Neural Networks (CNN), with Moth Flame Optimization (MFO) and Salp Swarm Optimization (SSO) algorithms to enhance lung cancer detection in CT images. Wavelet preprocessing extracts features, and MFO optimizes CNN hyperparameters. SSO refines feature selection for better classification accuracy, leading to more precise detection of malignant and benign nodules.

The system integrates a gesture-based interface for hands-free operation, streamlining interactions for medical professionals. MFO and SSO optimization enhance CNN performance, improving detection accuracy with fewer errors. This solution is cost-effective, providing rapid, reliable results that aid in early lung cancer diagnosis, optimizing workflow efficiency and improving patient care.

3.3 BLOCK DIAGRAM FOR PROPOSED SYSTEM

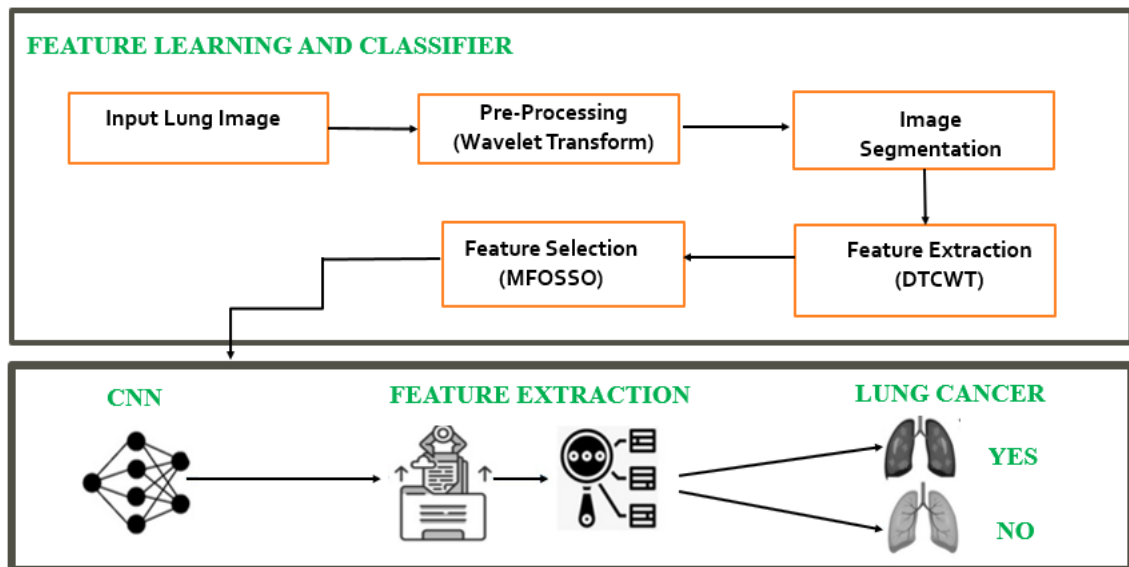


Fig.3.1- Block Diagram

3.4 APPROACH USED

Instead of relying solely on random data, integrate a more sophisticated sensor data simulation that mimics real-world scenarios. This could involve generating data patterns that reflect different types of accidents or incidents. If possible, incorporate real sensor data from physical sensors (temperature, distance, accelerometers) to provide more accurate and meaningful simulation. This would make the system more realistic and applicable to actual scenarios. Extend the alerting mechanism beyond email. Integrate SMS notifications, push notifications to mobile devices, or even automate alerts to emergency services through APIs. This ensures that alerts reach users promptly through more channels. This project employs image preprocessing with wavelet transforms and normalization, followed by feature extraction using Convolutional Neural Networks (CNN). Moth Flame Optimization (MFO) and Salp Swarm Optimization (SSO) are used for feature selection, enhancing classification performance. Additionally, a gesture-based control system using hand gestures is integrated for hands-free operation, allowing users to interact with the system intuitively. Evaluation is based on accuracy, AUC, and efficiency metrics.

3.5 PROCESS CYCLE

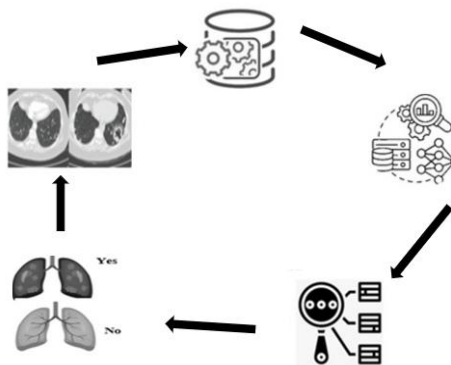


Fig.3.2 –PROCESSING OVERVIEW

3.6 FLOWCHART

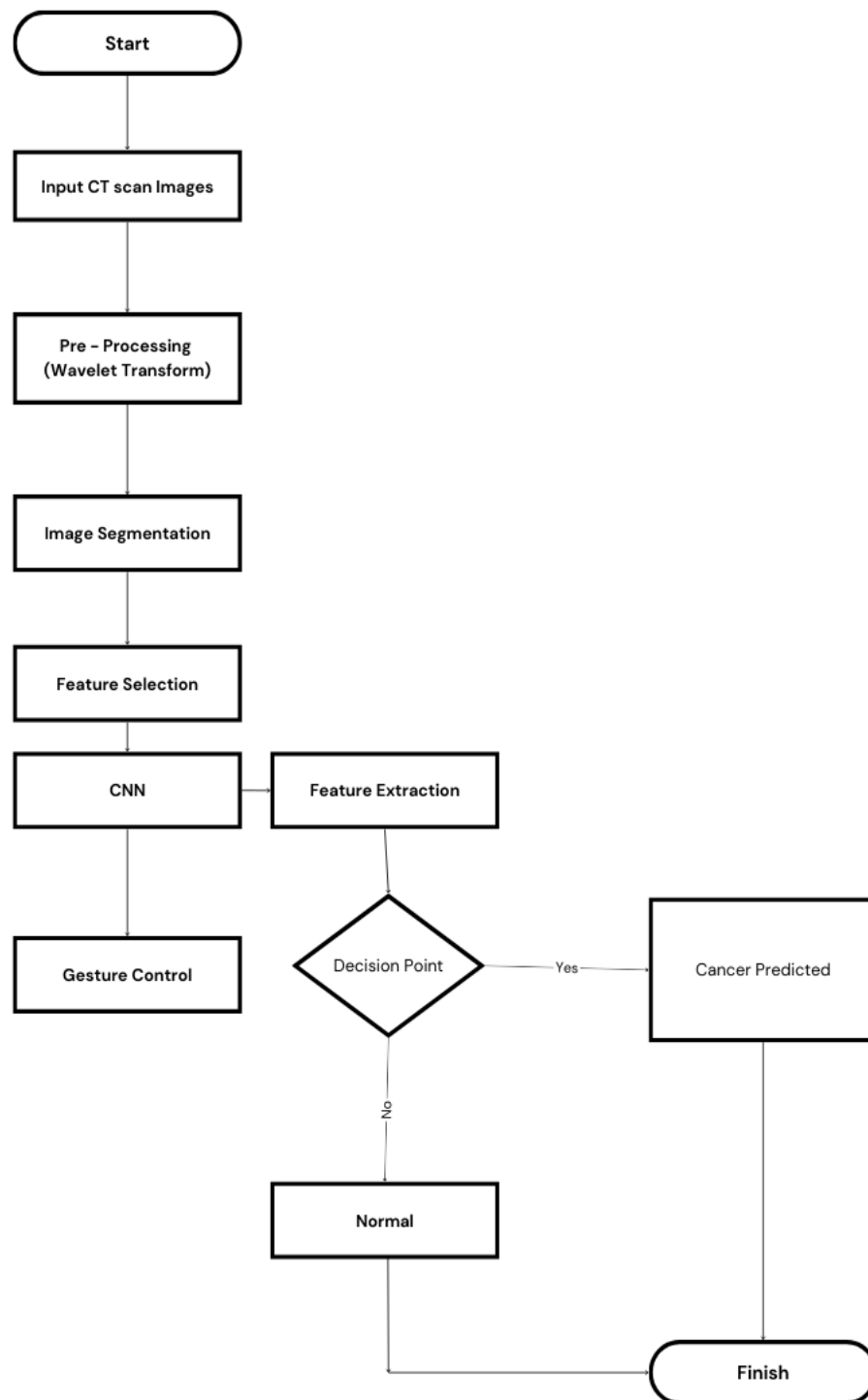


Fig.3.3 –PROCESS AND RESULT

3.7 USECASE - DIAGRAM

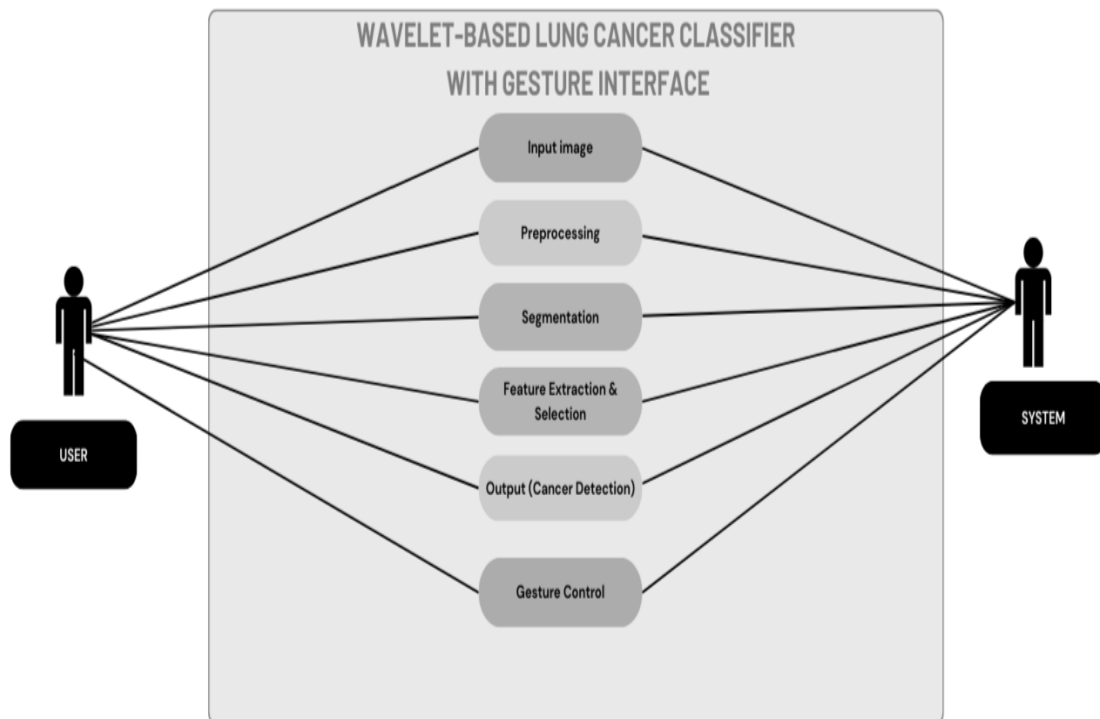


Fig.3.4-FUNCTIONS INVOLVED AND ACTIONS

CHAPTER 4

MODULES

4.1 MODULE DESCRIPTION

- Preprocessing
- Image Segmentation
- Feature Selection
- Classification
- Gesture Control
- Decision – Making

4.1.1. PREPROCESSING

The preprocessing module plays a critical role in ensuring the quality and usability of CT scan images for accurate lung cancer detection. This step involves wavelet-based transformation techniques that enhance image features while reducing noise and irrelevant data. The wavelet transform decomposes the CT scan images into various frequency components, enabling the extraction of significant features while preserving spatial information. This approach enhances the visibility of nodules and other abnormalities, facilitating better analysis in subsequent stages. Additionally, image normalization is applied to standardize intensity values, ensuring consistency across diverse datasets.

To further improve image quality, histogram equalization techniques are employed to enhance contrast, ensuring that critical details such as edges and textures are more distinguishable. This module also includes resizing and alignment of the images to ensure compatibility with the machine learning model's input requirements. Furthermore, artifacts and distortions commonly present in raw CT images are reduced using advanced denoising algorithms.

By optimizing image clarity and ensuring uniformity, the preprocessing module creates a robust foundation for accurate segmentation, feature extraction, and classification in the lung cancer detection system.

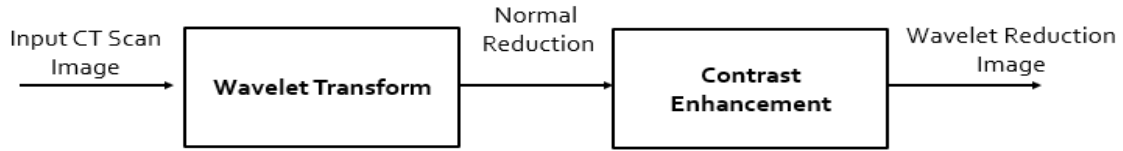


Fig.4.1- Functions of Preprocessing

4.1.2. IMAGE SEGMENTATION

Image segmentation is a crucial step in lung cancer detection, where the goal is to isolate lung regions and identify areas of interest, such as nodules, from CT scan images. This process separates the meaningful regions of an image from irrelevant background data, ensuring accurate analysis. In this project, segmentation begins with adaptive thresholding to distinguish lung structures based on intensity levels. This method dynamically adjusts thresholds, effectively separating nodules of varying sizes and densities from other lung tissues.

The segmentation process also employs marker-controlled watershed algorithms, which refine the boundaries of segmented regions by leveraging marker points to guide separation. Accurate segmentation improves the subsequent steps, such as feature extraction and classification, by isolating relevant areas and reducing computational complexity, thereby increasing the system's overall performance and diagnostic precision.

4.1.3. FEATURE SELECTION

Feature selection is a vital step in lung cancer detection that focuses on identifying the most relevant and significant features from CT scan images. It reduces the dimensionality of data by selecting key attributes such as nodule size, texture, intensity, and shape while discarding irrelevant or redundant features.

This improves the computational efficiency and ensures that only meaningful data is passed to the classification stage. In this project, advanced optimization techniques, including Moth Flame Optimization (MFO) and Salp Swarm Optimization (SSO), are utilized to prioritize features critical for accurate lung cancer prediction.

By emphasizing the most informative features, the feature selection module enhances the accuracy and robustness of the machine learning algorithms employed in the system. It mitigates the risk of overfitting and accelerates the training process, allowing the classifiers to focus solely on the significant patterns within the data. This step also plays a crucial role in ensuring the system's adaptability to diverse datasets by selecting features that generalize well across different cases. The combined use of MFO and SSO ensures an optimized feature set, maximizing prediction accuracy and improving the system's overall performance.

4.1.4. CLASSIFICATION

The classification module is a critical component of the lung cancer detection system, responsible for categorizing the segmented and feature-selected data into appropriate classes, such as benign or malignant. It uses machine

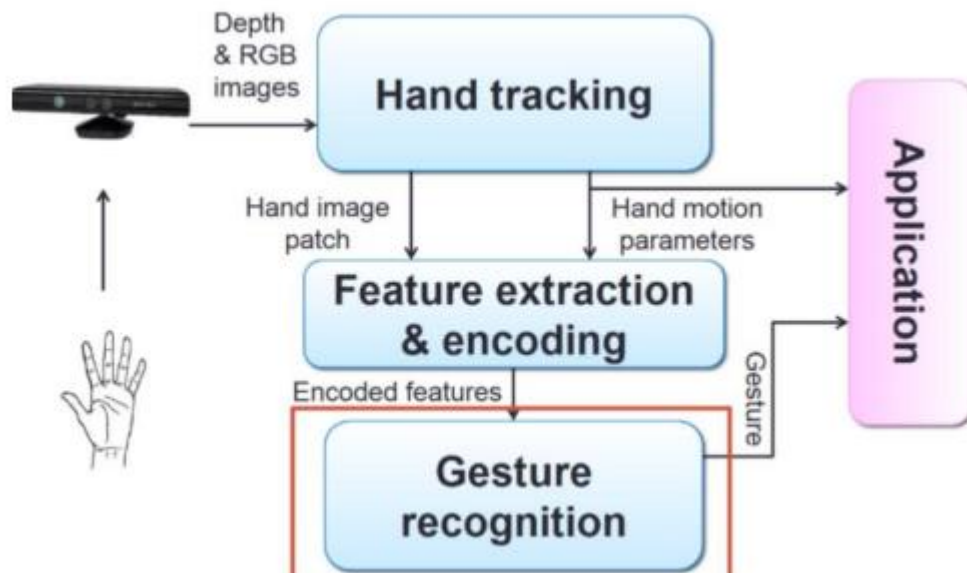
learning algorithms, including Convolutional Neural Networks (CNN), to analyze the extracted features from CT scan images. These algorithms learn to distinguish between cancerous and non-cancerous patterns by training on labeled datasets. The system continuously refines its classification abilities through backpropagation and optimization techniques, improving its prediction accuracy over time.

In this project, the CNN algorithm is employed for its ability to capture spatial hierarchies in image data, ensuring accurate identification of tumor characteristics. The classification process is bolstered by integrating optimization techniques like Moth Flame Optimization (MFO) and Salp Swarm Optimization (SSO), which enhance the performance of the CNN by optimizing the weight and feature selection. The classification module not only ensures fast and reliable predictions but also supports real-time diagnostic assistance for medical professionals, contributing to early and accurate lung cancer detection.

4.1.5. GESTURE CONTROL

The Gesture Control module in the lung cancer detection system enables hands-free operation, offering a user-friendly interface for healthcare professionals. It leverages computer vision techniques to recognize and interpret specific hand gestures, allowing users to navigate the system without physical interaction. This feature enhances the ease of use, especially in sterile environments such as hospitals, where minimizing direct contact with devices is essential for maintaining hygiene and safety.

This module employs techniques such as hand tracking and motion recognition, which are processed through deep learning models to detect and respond to user gestures. For example, simple hand movements can trigger the system to move through different stages of the detection process, such as inputting CT scan images, selecting features, or viewing results.



4.1.6. DECISION MAKING

as Support Vector Machine (SVM) and Random Forest, to evaluate the probabilities and ensure accurate predictions.

The system employs a decision threshold that defines the classification criteria for benign and malignant nodules based on the model's output. If the probability of malignancy exceeds a certain threshold, the system categorizes the nodule as malignant; otherwise, it is classified as benign. By using machine learning techniques, this module ensures that even subtle features or patterns not immediately visible to a human expert are accounted for, leading to a more reliable diagnosis.

Additionally, the Decision-Making module integrates the gesture control interface, allowing radiologists or healthcare professionals to interact with the system hands-free. This intuitive control enhances the user experience and provides an efficient way to navigate through the system. Ultimately, the decision-making process ensures that patients receive accurate, timely information about their condition, which can lead to faster interventions and improved treatment outcomes.

4.2 CNN ALGORITHM

Convolutional Neural Networks (CNNs) are a vital component of your lung cancer detection system, especially for image classification tasks. CNNs are designed to automatically and adaptively learn spatial hierarchies of features from input images, such as CT scan slices. In this module, the raw CT scan images are fed into the CNN architecture, which is designed to extract meaningful features from the images without needing manual feature extraction. The CNN model performs

well with high-dimensional data, such as images, where it can detect patterns, textures, and shapes related to lung cancer nodules.

4.2.1 FEATURE EXTRACTION PHASE

CNN primarily automates the feature extraction process. This phase involves:

- **Convolutional Layer:** Extract spatial and hierarchical features from lung CT images, identifying patterns like edges, textures, and structures.
- **Pooling Layers:** Reduce the spatial dimensions while preserving essential features, helping the model focus on critical areas such as nodules.

4.2.2 CLASSIFICATION PHASE

The extracted features are fed into fully connected layers to predict outcomes:

- CNN classifies the CT images into categories (e.g., benign or malignant).
- It outputs the probability of each class, ensuring an accurate and data-driven decision-making process.

These phases help ensure efficient and accurate lung cancer detection in your project.

4.3 HAAR CASCADE ALGORITHM

Haar Cascade algorithm is a machine learning-based object detection method widely used for identifying specific objects within images. It utilizes Haar-like features, which are simple rectangular features that capture essential information about the intensity variations in an image, such as edges and textures. This algorithm applies these features in a sliding window manner across an image at different scales

to detect objects of interest, including faces, eyes, or abnormalities in medical images like lung cancer nodules.

The algorithm is built on the concept of cascading classifiers. Initially, a simple classifier is used to filter out large portions of an image that do not contain the target object. Then, subsequent classifiers with more complex features are applied to refine the search. Each classifier in the cascade progressively becomes more complex, thus improving detection accuracy while ensuring fast processing. The process is highly efficient for real-time applications as it eliminates unnecessary computations at early stages, focusing only on regions that are more likely to contain the object of interest.

In the context of lung cancer detection, the Haar Cascade algorithm can be trained to detect lung tumors by using a large dataset of CT scan images. During training, the algorithm learns to distinguish between benign and malignant nodules based on the intensity patterns captured by the Haar-like features. This enables the model to classify the regions of CT scan images that potentially represent cancerous tissues.

One of the major advantages of the Haar Cascade algorithm is its ability to perform real-time object detection, which is particularly beneficial in time-sensitive applications like medical diagnostics. the Haar Cascade algorithm is often used in combination with other techniques, such as deep learning or pre-processing methods, to improve overall detection accuracy and handle the variability of medical imaging data.

CHAPTER 5

SYSTEM SPECIFICATION

5.1 SOFTWARE REQUIREMENT

- Google Colab
- Google Drive
- Open CV
- Media pie
- PyAutoGUI

5.1.1. GOOGLE COLAB

Google Colab is an essential tool for machine learning projects, offering a cloud-based environment for coding and model training without the need for local resources. In your project, Google Colab helps you run your lung cancer detection system efficiently by leveraging the power of free GPUs and TPUs. The environment is well-suited for deep learning, offering easy integration with libraries such as TensorFlow, Keras, and OpenCV, which are essential for image processing and classification tasks.

With Google Colab, you can upload and preprocess large datasets, including CT scan images, for training models. Colab provides seamless integration with Google Drive, which makes it easy to store, access, and share data and models. This is especially useful in research-based projects like lung cancer detection, where datasets can be large and require substantial storage. The flexibility of Google Colab ensures that the entire machine learning pipeline—

data preparation, model training, evaluation, and prediction—can be done collaboratively, without the burden of maintaining a local environment.

The interactive environment of Google Colab allows real-time execution of code and immediate feedback on the performance of the lung cancer detection model. You can implement machine learning algorithms like CNNs, optimize them using methods like Moth Flame Optimization (MFO) and Salp Swarm Optimization (SSO), and visualize results directly. The platform's ability to support different machine learning frameworks and libraries, coupled with cloud-based execution, enables rapid experimentation and model fine-tuning for accurate predictions, making it an invaluable tool for your project.

5.1.2. GOOGLE DRIVE

Google Drive plays a critical role in storing and managing large datasets for machine learning projects. In the context of your lung cancer detection system, Google Drive provides a centralized space to securely store CT scan images, model files, and other project resources. Since Google Drive offers substantial free storage and seamless integration with Google Colab, it becomes an invaluable tool for accessing and sharing data. This ensures that large datasets, which may not fit locally on your machine, are easily accessible for training, testing, and deployment of the machine learning model.

Google Drive also facilitates collaboration by allowing multiple users to access and work on the same project. Researchers and team members can upload and share resources like annotated datasets, pre-trained models, and experimental results without worrying about data transfer or versioning issues. Furthermore,

by connecting Google Drive with Google Colab, you can directly load and save files between the two platforms. This integration simplifies workflow management, enabling you to focus on building and refining your lung cancer detection model with minimal technical barriers.

5.1.3. OPEN CV

OpenCV (Open Source Computer Vision Library) is a powerful, open-source library used for computer vision and image processing tasks. In the context of your lung cancer detection project, OpenCV is crucial for handling CT scan image preprocessing, segmentation, and feature extraction. It provides a range of functions to process and manipulate images, including operations such as resizing, filtering, edge detection, and image transformations. By using OpenCV, you can prepare CT scan images for machine learning algorithms, enhance image quality, and extract relevant features that will be used in the classification process.

Moreover, OpenCV is also beneficial for integrating the gesture control module in your project. Through its extensive support for real-time image and video processing, OpenCV can track and recognize hand gestures by analyzing webcam inputs. By leveraging OpenCV's computer vision capabilities, users can interact with the system without the need for physical input devices, making the system more accessible and user-friendly. This allows for hands-free operation, improving the system's efficiency and usability in clinical environments.

5.1.4. MEDIA PIPE

In the realm of eye-controlled virtual mouse systems, MediaPipe emerges as a transformative technology, offering a versatile framework for developing real-time perception pipelines that augment the capabilities of such innovative interfaces. Developed by Google, MediaPipe provides a comprehensive solution for building machine learning-based pipelines for various perception tasks, including pose estimation, hand tracking, facial features detection, and object recognition.

By leveraging MediaPipe's modular architecture, developers can seamlessly integrate pre built components called "graphs" to construct custom pipelines tailored to the unique requirements of eye-controlled virtual mouse systems, enabling precise and intuitive interaction experiences. MediaPipe's extensive library of pre-trained models and algorithms significantly enhances the capabilities of eye-controlled virtual mouse systems, empowering developers to implement features such as gaze estimation, eye tracking, and blink detection with unparalleled accuracy and reliability. These capabilities form the foundation for enabling seamless cursor control, command execution, and interaction with on-screen elements solely through the user's gaze, facilitating intuitive and efficient user interfaces across diverse applications and use cases.

Moreover, MediaPipe's support for real-time processing is paramount in the context of eye-controlled virtual mouse systems, ensuring immediate responsiveness and seamless interaction with digital content. By leveraging MediaPipe's optimized algorithms and hardware acceleration features,

developers can achieve low-latency processing, enabling fluid cursor movements, precise gesture recognition, and immersive interaction experiences in real-time. Additionally, MediaPipe's support for multi-platform deployment ensures broad accessibility and usability across a diverse range of devices, including desktop computers, laptops, smartphones, and embedded systems, further enhancing the reach and impact of eye-controlled virtual mouse systems.

5.1.5. PyAutoGUI

PyAutoGUI, a Python library for controlling the mouse and keyboard, can be seamlessly integrated into an eye-controlled virtual mouse system to enhance its functionality and usability. By combining PyAutoGUI with eye-tracking technology, developers can create a more intuitive and efficient interaction experience for users. For instance, PyAutoGUI can be utilized to translate the user's eye movements into cursor movements on the screen, allowing them to navigate interfaces, click on elements, and interact with applications solely through their gaze. Moreover, PyAutoGUI's capabilities extend beyond basic cursor control; it enables developers to automate repetitive tasks, simulate keyboard inputs, and interact with GUI elements programmatically. This integration opens up a myriad of possibilities for enhancing accessibility, improving productivity, and enabling new modes of interaction in various domains, including assistive technology, gaming, and human-computer interaction research.

With PyAutoGUI's user-friendly interface and extensive documentation, developers can easily implement sophisticated features in eye-controlled virtual

mouse systems, ultimately enhancing the user experience and expanding the utility of such innovative technologies. Furthermore, PyAutoGUI offers features such as screen capturing and image recognition, which can be leveraged to enhance the functionality of an eye-controlled virtual mouse. Developers can utilize screen capturing to analyze the content displayed on the screen, enabling context-aware interactions based on the visual elements present. Additionally, image recognition capabilities allow the system to identify specific on-screen objects or buttons, enabling more precise and targeted interactions.

This advanced functionality opens up possibilities for creating custom commands triggered by specific visual cues, improving the overall efficiency and usability of the eye-controlled virtual mouse system. With PyAutoGUI's versatile capabilities seamlessly integrated with eye-tracking technology, developers can innovate and create immersive, intuitive, and highly efficient user interfaces tailored to individual needs.

5.2 HARDWARE REQUIREMENT

- Processor – Intel i3 or Higher.
- RAM – 4GB or Higher.
- Storage – 150GB or Higher.

CHAPTER 6

METHODOLOGY

6.1. DATA COLLECTION AND DATASET PREPARATION

The first step in the methodology is data collection. The system relies on a comprehensive dataset of lung CT scan images, which are collected from reputable public medical repositories such as the LIDC-IDRI (Lung Image Database Consortium and Image Database Resource Initiative) or similar sources. These datasets include labeled data, with CT scans marked as either benign or malignant, providing a robust foundation for training and testing the system. Along with the CT scans, additional patient information, such as age, smoking history, and comorbidities, can be incorporated to improve prediction accuracy.

The dataset is then split into two subsets: a training set and a testing set. The training set is used to train the model, while the testing set is used to evaluate its performance. This separation ensures that the model is not overfitted to the training data and provides a more realistic evaluation of how it will perform on new, unseen data. Furthermore, the data is pre-processed to ensure consistency across images. This includes resizing all images to a standard dimension and performing any necessary transformations to normalize pixel values for consistent representation.

6.2. PREPROCESSING OF CT SCAN IMAGES

In the preprocessing stage, the raw CT scan images undergo various transformations to improve their quality and remove noise. The CT scans can have inconsistencies in terms of contrast, brightness, and noise, all of which can hinder the performance of machine learning models. To mitigate this, image

normalization is applied to standardize pixel intensity values. A typical approach involves scaling pixel values to fall within a specific range, such as between 0 and 1. This helps the model better handle variations in image brightness. Additionally, noise reduction techniques like Gaussian smoothing or median filtering are applied to reduce irrelevant pixel variations.

This step is crucial as CT scans can have artifacts caused by equipment limitations or patient movement, which may reduce the model's ability to make accurate predictions. After noise reduction, image segmentation is carried out, where key regions of interest (such as the lungs and potential nodules) are extracted from the image. Segmentation algorithms, such as adaptive thresholding and marker-controlled watershed, are used to isolate the lung regions, helping to focus the model's attention on the most important features.

6.3. FEATURE EXTRACTION

Feature extraction is an essential step in converting raw CT scan data into a format that can be processed by machine learning algorithms. In this step, the system analyzes the segmented images and identifies important characteristics, such as shape, texture, intensity, and statistical properties of the lung nodules. Different types of feature extraction methods are employed to capture the key properties that can differentiate benign from malignant tumors.

Wavelet transform is used to extract multi-resolution features from the CT scan images. This allows the model to capture both fine-grained and high-level patterns in the data, which are important for accurate detection.

6.4. FEATURE SELECTION

The extracted features are then processed through a feature selection step to improve the system's performance. Feature selection is crucial because not all extracted features contribute equally to the classification task. Some features may be redundant or irrelevant, and their presence can increase the model's complexity and reduce computational efficiency. Therefore, the feature selection process aims to identify the most significant features that contribute to distinguishing between benign and malignant nodules.

In this project, advanced optimization algorithms like Moth Flame Optimization (MFO) and Salp Swarm Optimization (SSO) are used to perform feature selection. MFO and SSO are both nature-inspired algorithms that mimic the behaviors of moths and salps, respectively, to optimize solutions. These algorithms evaluate the importance of each feature by assessing its contribution to the classification accuracy. The optimal set of features selected by these algorithms is then passed to the classification stage. This step enhances the system's efficiency and accuracy by reducing the number of features, thereby decreasing the risk of overfitting and improving the model's generalization.

6.5. CLASSIFICATION USING DEEP LEARNING

After feature selection, the processed features are fed into a machine learning model for classification. A Convolutional Neural Network (CNN) is chosen due to its exceptional ability to process image data and capture hierarchical features from raw pixel data. The CNN model consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers. The convolutional layers

automatically learn spatial hierarchies of features, while the pooling layers reduce the dimensionality and computational complexity of the network.

The CNN model is trained on the feature vectors extracted from the CT scan images, using the training dataset. During training, the network adjusts its internal parameters (weights and biases) to minimize the error between the predicted and actual outcomes using backpropagation. Once trained, the model can classify new CT scan images into categories such as benign or malignant, based on the learned features. The classification accuracy is evaluated using the testing dataset, and the model's performance is optimized through iterative training.

6.6. GESTURE CONTROL INTEGRATION

A unique feature of this system is the integration of gesture control, which enables hands-free operation. The system incorporates gesture recognition to allow medical professionals to interact with the lung cancer detection system without touching the device. This is achieved using the MediaPipe library, which detects hand movements and translates them into specific actions, such as scrolling through CT scan images, zooming in/out, or activating different functions.

For instance, a user can swipe left or right to navigate between images, or use a fist gesture to zoom into an area of interest. This gesture-based interaction is particularly beneficial in medical environments, where professionals may need to maintain sterility or may not have access to a mouse or keyboard. The integration of gesture control improves workflow efficiency, enabling healthcare professionals to work faster while reducing the need for manual input.

6.7. POST-PROCESSING AND DECISION MAKING

Once the CT scan images have been classified into benign or malignant categories, the system outputs the prediction result to the user. However, to ensure that the diagnosis is reliable and accurate, post-processing steps are employed. These steps include generating confidence scores for each prediction, which represent the certainty of the classification. If the confidence score is below a certain threshold, the system may prompt the user to review the result.

Additionally, decision-making algorithms may be used to integrate the classification results with other patient information, such as their medical history or demographic details. This step provides more context to the lung cancer diagnosis and assists clinicians in making informed decisions. The system may also generate visual aids like heatmaps to highlight areas of the CT scan that contributed to the classification, helping doctors better interpret the results. This visual feedback ensures that medical professionals are confident in their decision-making.

6.8. SYSTEM INTEGRATION AND TESTING

Once all the individual modules, including data collection, preprocessing, feature extraction, feature selection, classification, gesture control, and decision-making, have been developed and tested separately, they are integrated into a single cohesive system. The system is tested for overall functionality, ensuring that all modules work seamlessly together. During this phase, edge cases and real-world data are used to test the robustness of the system.

The integrated system undergoes rigorous evaluation to assess its usability, speed, and reliability. The testing phase also includes gathering feedback from

medical professionals to ensure that the system meets clinical standards and is easy to use. Any issues identified during testing are addressed by iterating on the system and improving its performance.

6.9. EVALUATION AND OPTIMIZATION

After the system is fully integrated, it undergoes a series of evaluations to measure its performance. Key evaluation metrics such as accuracy, sensitivity, specificity, precision, and F1-score are used to assess how well the system classifies CT scan images. The performance of the system is benchmarked against existing methods, and any areas where it underperforms are identified.

Optimization techniques such as hyperparameter tuning, fine-tuning the MFO and SSO algorithms for feature selection, and adjusting the CNN architecture are applied to improve performance. Additionally, the system is evaluated for computational efficiency to ensure that it can process CT scans in real-time, which is critical for clinical use. Continuous testing and optimization are performed to ensure that the system can maintain high accuracy and efficiency as it is used in real-world settings.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

In this project, a comprehensive and efficient lung cancer detection system was developed using advanced machine learning techniques and optimization algorithms. By leveraging the capabilities of Moth Flame Optimization (MFO) and Salp Swarm Optimization (SSO) for feature selection, the system significantly enhanced the accuracy and efficiency of detecting malignant nodules from CT scan images. The integration of deep learning methods, particularly Convolutional Neural Networks (CNN), ensured precise and reliable classification, making the system robust for practical medical applications.

The incorporation of gesture control using MediaPipe added a unique dimension to the project, enabling hands-free interaction for healthcare professionals. This feature ensures seamless operation in sterile medical environments and improves workflow efficiency. The system also included a user-friendly interface and real-time processing capabilities, making it a practical tool for radiologists and clinicians. The modular structure of the system allows for future enhancements and adaptability to other diagnostic applications.

In conclusion, this project represents a significant advancement in the early detection of lung cancer, providing a cost-effective, efficient, and accurate diagnostic tool. By addressing the limitations of existing methods and integrating innovative technologies, the system has the potential to improve patient outcomes and streamline medical workflows.

7.2 FUTURE ENHANCEMENT

One of the major future directions for this system is the integration with cloud-based platforms to enable remote access and scalability. Such a feature would allow clinicians and researchers to interact with the system from any location, facilitating telemedicine and global collaborations. Real-time data integration, including patient medical histories and vital signs, could further enhance diagnostic accuracy, offering a more comprehensive analysis of lung cancer cases.

Another promising enhancement involves expanding the system to detect multiple forms of cancer. By training the model on diverse datasets, it can be adapted to classify and diagnose other cancers, increasing its utility in oncology. Additionally, advancements in gesture control technology, such as integrating voice commands or refining the recognition algorithms, would make the system more intuitive and accessible to healthcare professionals, especially in high-pressure clinical environments.

To improve diagnostic efficiency, future versions of the system can explore the adoption of more advanced machine learning architectures, such as transformer-based models, or hybrid optimization techniques. Furthermore, developing a mobile application could make this system portable and accessible for use in rural or under-resourced healthcare settings. Coupled with explainable AI features, such enhancements will foster trust among medical professionals by providing transparent insights into the decision-making process of the system.

APPENDIX – 1

SOURCE CODE

Lung Cancer Prediction.py

```
# Mount Google Drive to access dataset files
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

# Define paths to the training, validation, and test datasets
train_folder = '/content/drive/MyDrive/dataset/train'
test_folder = '/content/drive/MyDrive/dataset/test'
validate_folder = '/content/drive/MyDrive/datasetvalid'

# Define paths to the specific classes within the dataset
normal_folder = '/normal'
adenocarcinoma_folder = '/adenocarcinoma_left.lower.lobe_T2_N0_M0_Ib'
large_cell_carcinoma_folder = '/large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa'
squamous_cell_carcinoma_folder =
'/squamous.cell.carcinoma_left.hilum_T1_N2_M0_IIIa'

# Import necessary libraries
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn import datasets
from sklearn.model_selection import train_test_split
```

```

from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.preprocessing import LabelEncoder

import tensorflow as tf
import tensorflow.keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, SpatialDropout2D, Activation,
Lambda, Flatten, LSTM
from tensorflow.keras.layers import Conv2D, MaxPooling2D,
GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam, RMSprop
from tensorflow.keras import utils

print("Libraries Imported")

# Set the image size for resizing
IMAGE_SIZE = (350, 350)

# Initialize the image data generators for training and testing
train_datagen = ImageDataGenerator(rescale=1./255, horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)

# Define the batch size for training
batch_size = 8

# Create the training data generator
train_generator = train_datagen.flow_from_directory(
    train_folder,

```

```

    target_size=IMAGE_SIZE,
    batch_size=batch_size,
    color_mode="rgb",
    class_mode='categorical'
)

# Create the validation data generator
validation_generator = test_datagen.flow_from_directory(
    test_folder,
    target_size=IMAGE_SIZE,
    batch_size=batch_size,
    color_mode="rgb",
    class_mode='categorical'
)

# Set up callbacks for learning rate reduction, early stopping, and model checkpointing
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping,
ModelCheckpoint
learning_rate_reduction = ReduceLROnPlateau(monitor='loss',
patience=5, verbose=2, factor=0.5, min_lr=0.000001)
early_stops = EarlyStopping(monitor='loss', min_delta=0, patience=6, verbose=2,
mode='auto')
checkpointer=ModelCheckpoint(filepath='best_model.hdf5', verbose=2,
save_best_only=True, save_weights_only=True)

# Define the number of output classes
OUTPUT_SIZE = 4

# Load a pre-trained model (Xception) without the top layers and freeze its weights
pretrained_model=tf.keras.applications.Xception(weights='imagenet',
include_top=False, input_shape=[*IMAGE_SIZE, 3])
pretrained_model.trainable = False

```



```

# Create a new model with the pre-trained base and additional layers for classification
model = Sequential()
model.add(pretrained_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(OUTPUT_SIZE, activation='softmax'))

print("Pretrained model used:")
pretrained_model.summary()

print("Final model created:")
model.summary()

# Compile the model with an optimizer, loss function, and evaluation metric
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model with the training and validation data generators
history = model.fit(
    train_generator,
    steps_per_epoch=25,
    epochs=50,
    callbacks=[learning_rate_reduction, early_stops, checkpointer],
    validation_data=validation_generator,
    validation_steps=20
)
print("Final training accuracy =", history.history['accuracy'][-1])
print("Final testing accuracy =", history.history['val_accuracy'][-1])

# Function to display training curves for loss and accuracy
def display_training_curves(training, validation, title, subplot):
    if subplot % 10 == 1:
        plt.subplots(figsize=(10, 10), facecolor='#F0F0F0')

```

```

plt.tight_layout()
ax = plt.subplot(subplot)
ax.set_facecolor('#F8F8F8')
ax.plot(training)
ax.plot(validation)
ax.set_title('model ' + title)
ax.set_ylabel(title)
ax.set_xlabel('epoch')
ax.legend(['train', 'valid.'])

# Display training curves for loss and accuracy
display_training_curves(history.history['loss'], history.history['val_loss'], 'loss', 211)
display_training_curves(history.history['accuracy'], history.history['val_accuracy'],
'accuracy', 212)

# Save the trained model
model.save('/content/drive/MyDrive/dataset/trained_lung_cancer_model.h5')

# Function to load and preprocess an image for prediction
from tensorflow.keras.preprocessing import image
import numpy as np
def load_and_preprocess_image(img_path, target_size):
    img = image.load_img(img_path, target_size=target_size)
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0 # Rescale the image like the training images
    return img_array
# Load, preprocess, and predict the class of an image
img_path = '/content/sq.png'
img = load_and_preprocess_image(img_path, IMAGE_SIZE)
predictions = model.predict(img)

```

```

predicted_class = np.argmax(predictions[0])
class_labels = list(train_generator.class_indices.keys())
predicted_label = class_labels[predicted_class]
print(f"The image belongs to class: {predicted_label}")
# Display the image with the predicted class
plt.imshow(image.load_img(img_path, target_size=IMAGE_SIZE))
plt.title(f"Predicted: {predicted_label}")
plt.axis('off')
plt.show()
# Repeat the process for additional images
img_path = '/content/ad3.png'
img = load_and_preprocess_image(img_path, IMAGE_SIZE)
predictions = model.predict(img)
predicted_class = np.argmax(predictions[0])
predicted_label = class_labels[predicted_class]
print(f"The image belongs to class: {predicted_label}")
plt.imshow(image.load_img(img_path, target_size=IMAGE_SIZE))
plt.title(f"Predicted: {predicted_label}")
plt.axis('off')
plt.show()
img_path = '/content/l3.png'
img = load_and_preprocess_image(img_path, IMAGE_SIZE)
predictions = model.predict(img)
predicted_class = np.argmax(predictions[0])
predicted_label = class_labels[predicted_class]
print(f"The image belongs to class: {predicted_label}")
plt.imshow(image.load_img(img_path, target_size=IMAGE_SIZE))
plt.title(f"Predicted: {predicted_label}")
plt.axis('off')
plt.show()
img_path = '/content/n8.jpg'

```

```

img = load_and_preprocess_image(img_path, IMAGE_SIZE)
predictions = model.predict(img)
predicted_class = np.argmax(predictions[0])
predicted_label = class_labels[predicted_class]
print(f"The image belongs to class: {predicted_label}")
plt.imshow(image.load_img(img_path, target_size=IMAGE_SIZE))
plt.title(f"Predicted: {predicted_label}")
plt.axis('off')
plt.show()

```

main.py

```

import cv2

import mediapipe as mp

import pyautogui

cap = cv2.VideoCapture(0)

hand_detector = mp.solutions.hands.Hands()

drawing_utils = mp.solutions.drawing_utils

screen_width, screen_height = pyautogui.size()

index_y = 0

while True:

    _, frame = cap.read()

    frame = cv2.flip(frame, 1)

    frame_height, frame_width, _ = frame.shape

    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    output = hand_detector.process(rgb_frame)

    hands = output.multi_hand_landmarks

```

```

if hands:

    for hand in hands: drawing_utils.draw_landmarks(frame, hand)

    landmarks = hand.landmark

    for id, landmark in enumerate(landmarks):

        x = int(landmark.x*frame_width)

        y = int(landmark.y*frame_height)

        if id == 8:

            cv2.circle(img=frame, center=(x,y), radius=10, color=(0, 255, 255))

            index_x = screen_width/frame_width*x

            index_y = screen_height/frame_height*y

            if id == 4:

                cv2.circle(img=frame, center=(x,y), radius=10, color=(0, 255, 255))

                thumb_x = screen_width/frame_width*x

                thumb_y = screen_height/frame_height*y

                print('outside', abs(index_y - thumb_y))

                if abs(index_y - thumb_y) < 20:

                    pyautogui.click()

                    pyautogui.sleep(1)

                elif abs(index_y - thumb_y) < 100:

                    pyautogui.moveTo(index_x, index_y)

            cv2.imshow('Virtual Mouse', frame)

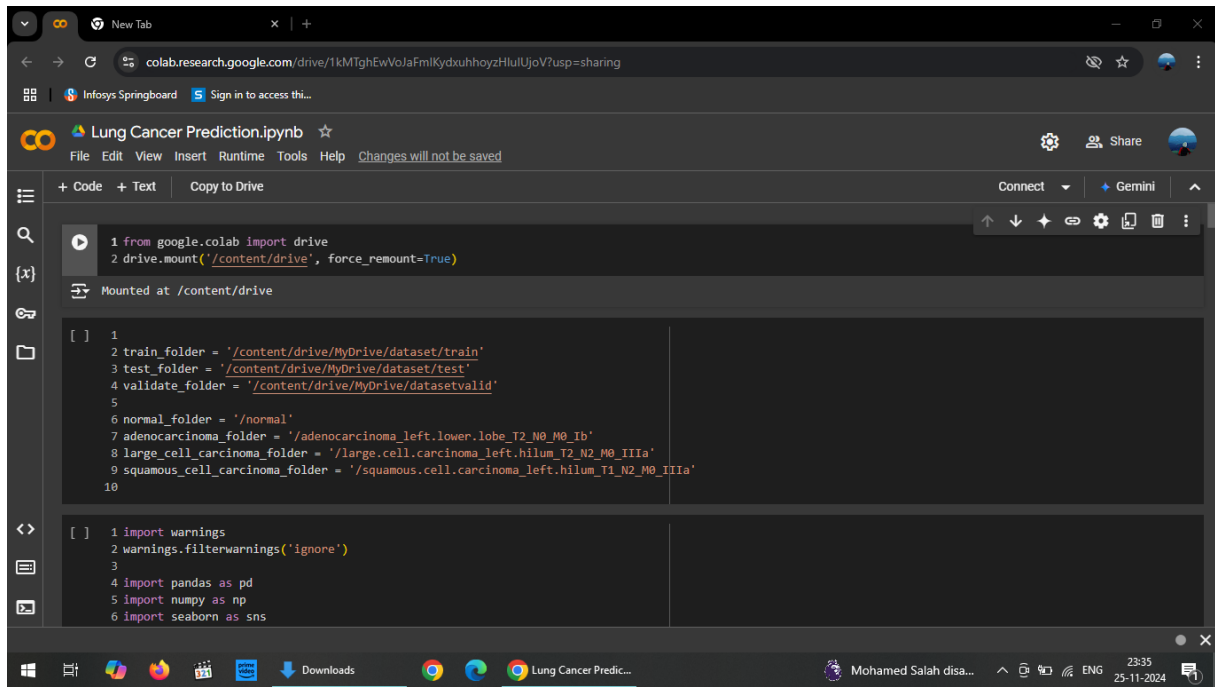
        cv2.waitKey(1)

```

APPENDIX – 2

SCREENSHOTS

Sample Output



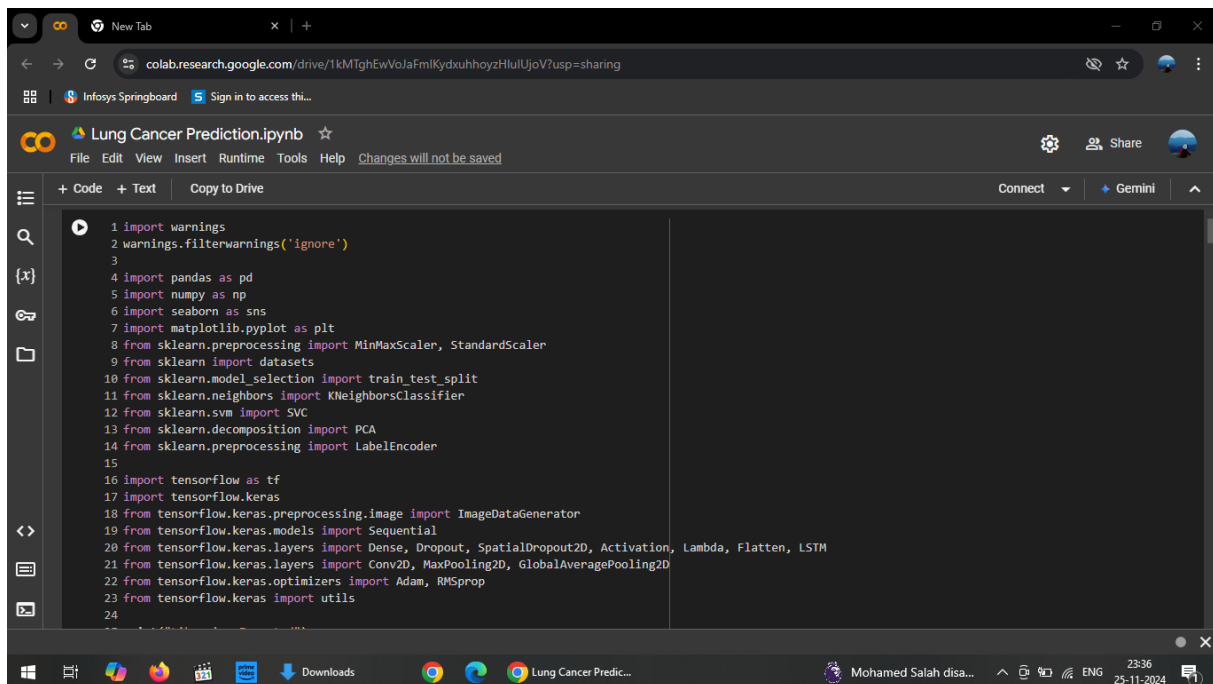
```
1 from google.colab import drive
2 drive.mount('/content/drive', force_remount=True)

Mounted at /content/drive

[ ] 1
2 train_folder = '/content/drive/MyDrive/dataset/train'
3 test_folder = '/content/drive/MyDrive/dataset/test'
4 validate_folder = '/content/drive/MyDrive/datasetvalid'
5
6 normal_folder = '/normal'
7 adenocarcinoma_folder = '/adenocarcinoma_left.lower.lobe_T2_N0_M0_Ib'
8 large_cell_carcinoma_folder = '/large.cell.carcinoma_left.hilum_T2_N2_M0_IIIIa'
9 squamous_cell_carcinoma_folder = '/squamous.cell.carcinoma_left.hilum_T1_N2_M0_IIIIa'
10

[ ] 1 import warnings
2 warnings.filterwarnings('ignore')
3
4 import pandas as pd
5 import numpy as np
6 import seaborn as sns
```

Fig.No.2.1: Execution of code



```
1 import warnings
2 warnings.filterwarnings('ignore')
3
4 import pandas as pd
5 import numpy as np
6 import seaborn as sns
7 import matplotlib.pyplot as plt
8 from sklearn.preprocessing import MinMaxScaler, StandardScaler
9 from sklearn import datasets
10 from sklearn.model_selection import train_test_split
11 from sklearn.neighbors import KNeighborsClassifier
12 from sklearn.svm import SVC
13 from sklearn.decomposition import PCA
14 from sklearn.preprocessing import LabelEncoder
15
16 import tensorflow as tf
17 import tensorflow.keras
18 from tensorflow.keras.preprocessing.image import ImageDataGenerator
19 from tensorflow.keras.models import Sequential
20 from tensorflow.keras.layers import Dense, Dropout, SpatialDropout2D, Activation, Lambda, Flatten, LSTM
21 from tensorflow.keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
22 from tensorflow.keras.optimizers import Adam, RMSprop
23 from tensorflow.keras import utils
24
```

Fig.No.2.2: Execution of code

```

18
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_dim_ordering_tf_kernels_notop.h5
83683744/83683744 [=====] - 3s 0us/step
Pretrained model used:
Model: "xception"

Layer (type)                 Output Shape          Param #   Connected to
-----
input_3 (InputLayer)         [(None, 350, 350, 3)] 0           []
block1_conv1 (Conv2D)         (None, 174, 174, 32)  864        ['input_3[0][0]']
block1_conv1_bn (BatchNorma (None, 174, 174, 32)  128        ['block1_conv1[0][0]']
block1_conv1_act (Activati (None, 174, 174, 32)  0           ['block1_conv1_bn[0][0]']
block1_conv2 (Conv2D)         (None, 172, 172, 64)  18432      ['block1_conv1_act[0][0]']
block1_conv2_bn (BatchNorma (None, 172, 172, 64)  256        ['block1_conv2[0][0]']
block1_conv2_act (Activati (None, 172, 172, 64)  0           ['block1_conv2_bn[0][0]']
block3_conv3 (Conv2D)         (None, 172, 172, 128) 8768        ['block1_conv2_act[0][0]']

```

Fig.No.2.3: Execution of code

```

1 history = model.fit(
2     train_generator,
3     steps_per_epoch=25,
4     epochs=50,
5     callbacks=[learning_rate_reduction, early_stops, checkpointer],
6     validation_data=validation_generator,
7     validation_steps=20
8 )
9
10 print("Final training accuracy =", history.history['accuracy'][-1])
11 print("Final testing accuracy =", history.history['val_accuracy'][-1])
12
Epoch 1/50
25/25 [=====] - ETA: 0s - loss: 1.2235 - accuracy: 0.4467
Epoch 1: val_loss improved from inf to 1.08683, saving model to best_model.hdf5
25/25 [=====] - 201s 8s/step - loss: 1.2235 - accuracy: 0.4467 - val_loss: 1.0868 - val_accuracy: 0.5437 - lr: 0.0010
Epoch 2/50
25/25 [=====] - ETA: 0s - loss: 1.0109 - accuracy: 0.5400
Epoch 2: val_loss improved from 1.08683 to 1.01950, saving model to best_model.hdf5
25/25 [=====] - 251s 10s/step - loss: 1.0109 - accuracy: 0.5400 - val_loss: 1.0195 - val_accuracy: 0.5250 - lr: 0.0010
Epoch 3/50
25/25 [=====] - ETA: 0s - loss: 0.8993 - accuracy: 0.6200
Epoch 3: val_loss improved from 1.01950 to 0.86878, saving model to best_model.hdf5
25/25 [=====] - 252s 10s/step - loss: 0.8993 - accuracy: 0.6200 - val_loss: 0.8688 - val_accuracy: 0.5938 - lr: 0.0010

```

Fig.No.2.4: Execution of code

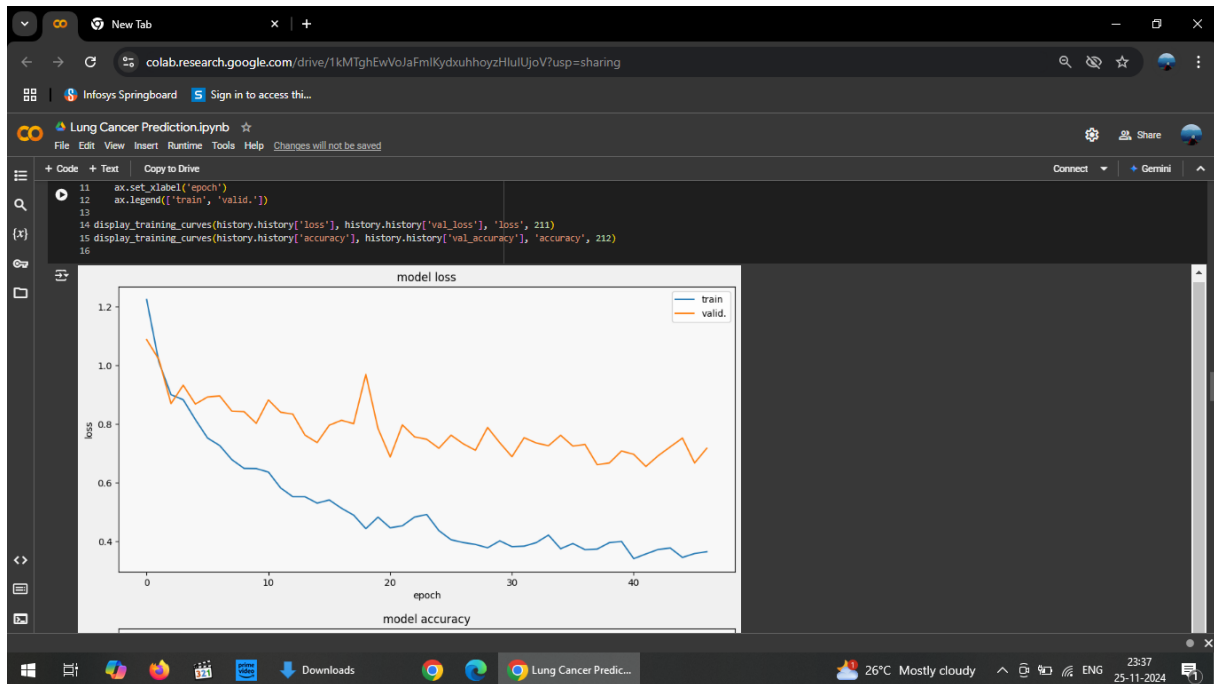


Fig.No.2.5: Accuracy Rate

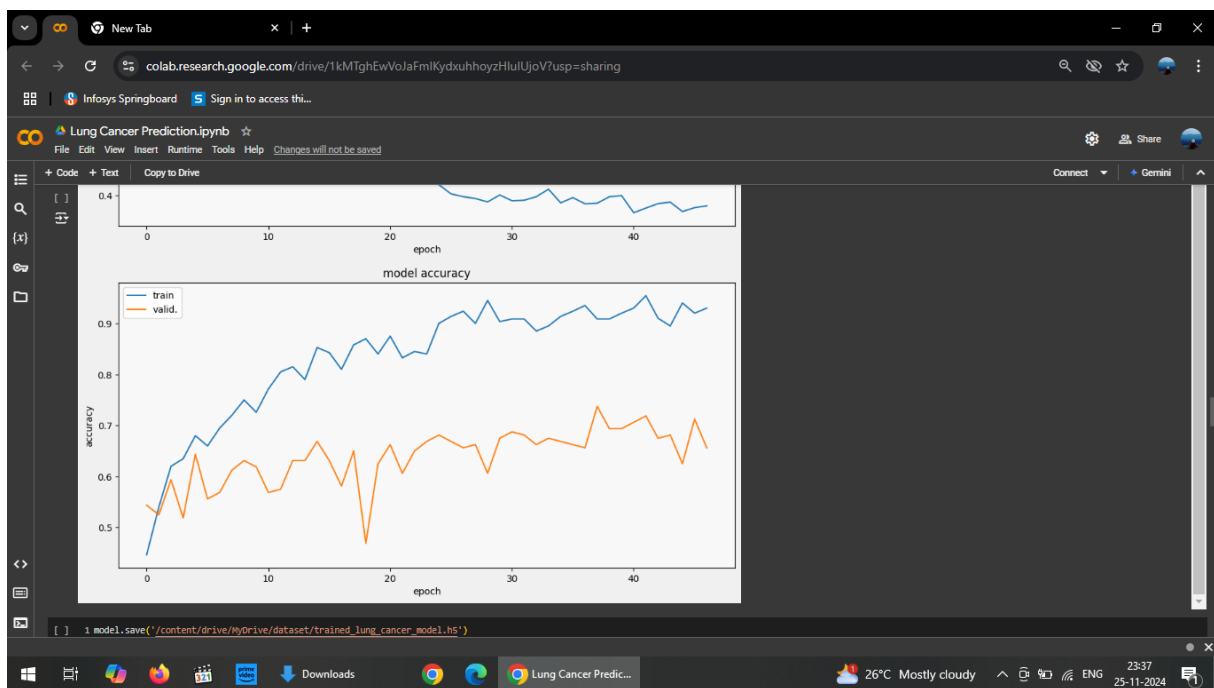


Fig.No.2.6: Accuracy Rate

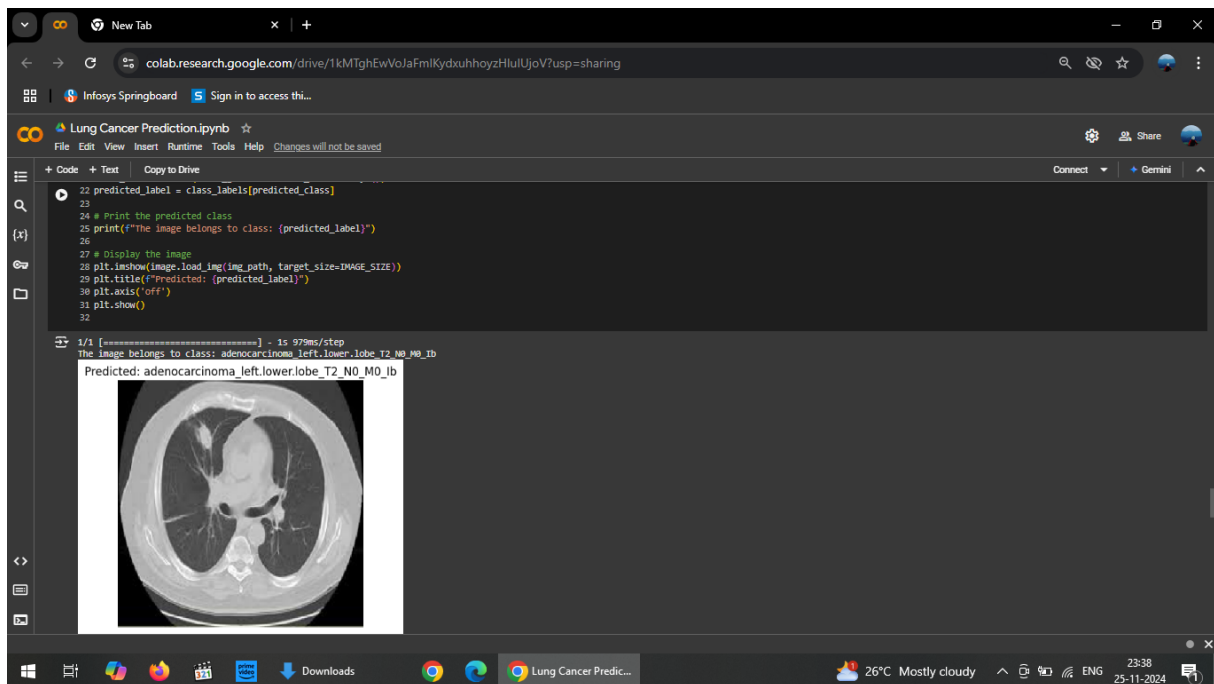


Fig.No.2.7: Predicted Image

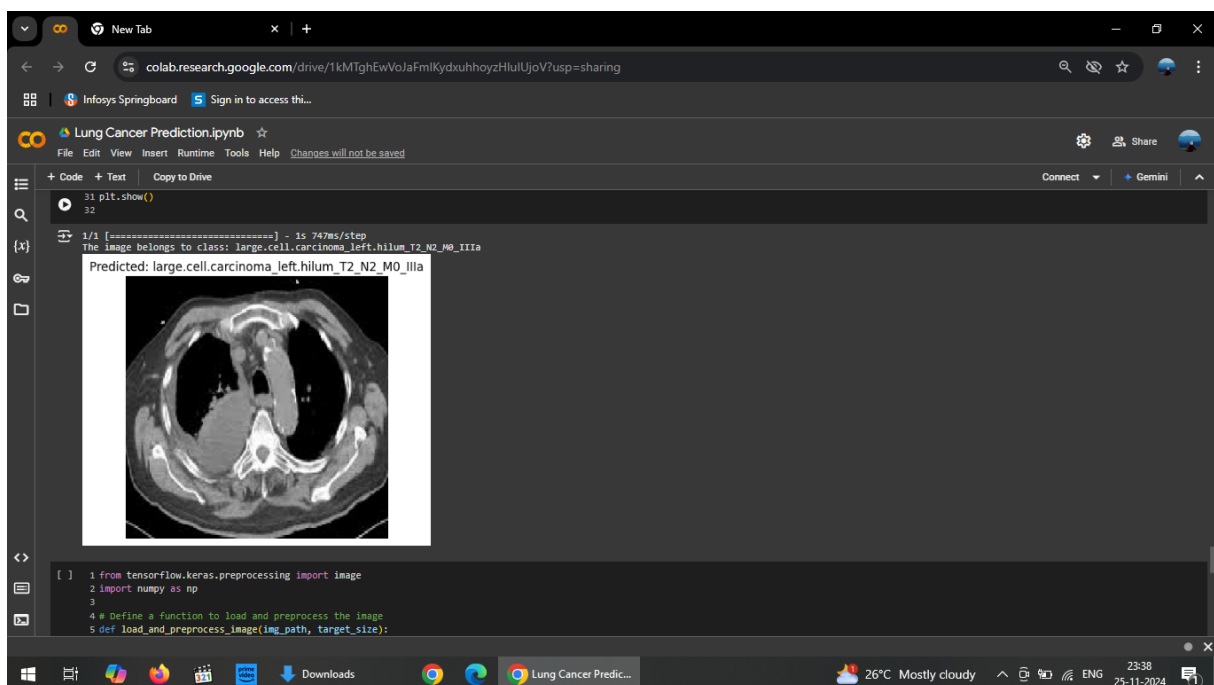


Fig.No.2.8: Predicted Image

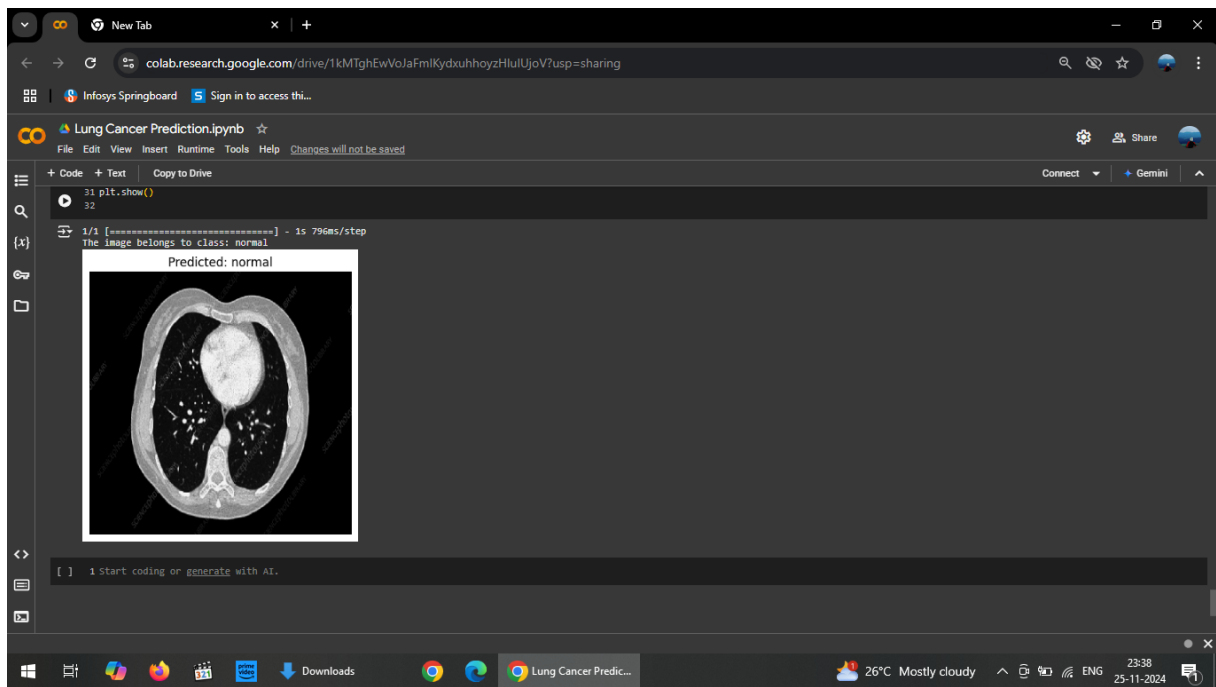


Fig.No.2.9: Predicted Image

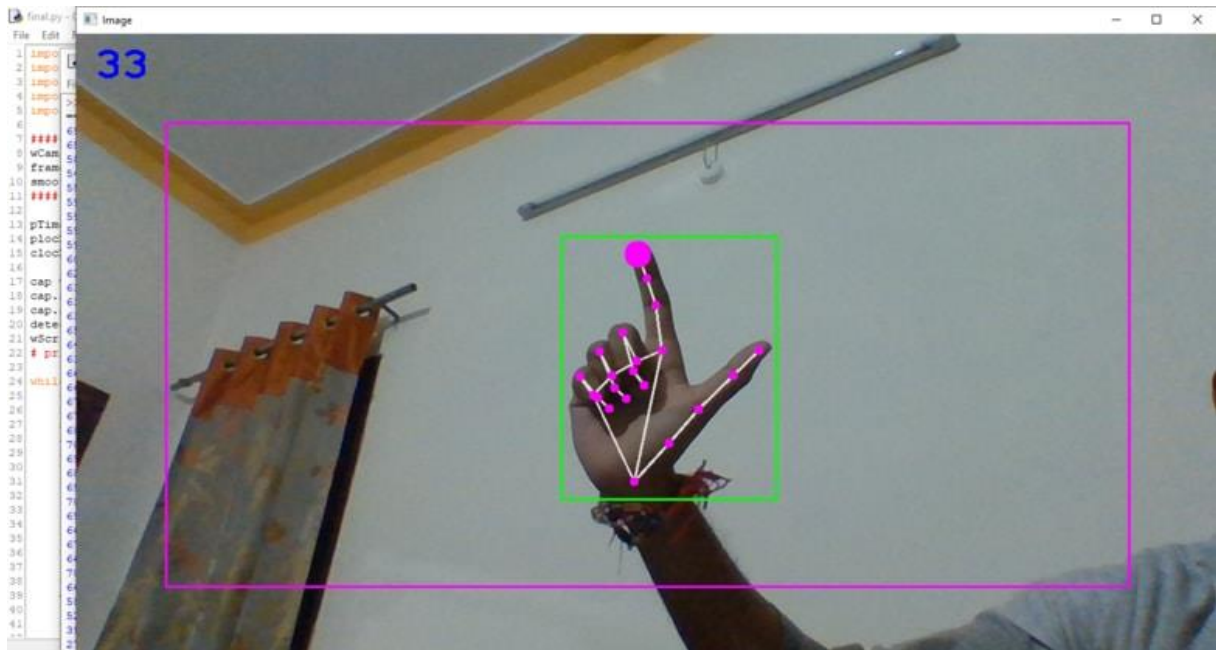


Fig.No.2.8: Gesture Control

REFERENCE

1. Y. Chen, Y. Wang, F. Hu and D. Wang, "A lung dense deep convolution neural network for robust lung parenchyma segmentation", IEEE Access, vol. 8, pp. 93527-93547, 2020.
2. B. S, P. R and A. B, "Lung Cancer Detection using Machine Learning," 2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC),Salem,India,2022 ,doi:10.1109/ICAAIC53929.2022.9793061.
3. R. An enhanced computer-assisted lung cancer detection method using content based image retrieval and data mining techniques B. Muthazhagan¹· T. Ravi· D.Rajiniginath ,Received: 5 March 2020 / Accepted: 14 May 2021
4. R. P.R., R. A. S. Nair and V. G., "A Comparative Study of Lung Cancer Detection using Machine Learning Algorithms," India, 2019, pp. 1-4, doi: 10.1109/ICECCT.2019.8869001.
5. R. Thriumani et al., "A preliminary study on in-vitro lung cancer detection using E-nose technology," 2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014), Penang, Malaysia, 2014, pp. 601-605, doi: 10.1109/ICCSCE.2014.7072789.
6. M. Ragab, I. Katib, S. A. Sharaf, F. Y. Assiri, D. Hamed and A. A. -M. Al-Ghamdi, "Self-Upgraded Cat Mouse Optimizer With Machine Learning Driven Lung Cancer Classification on Computed Tomography Imaging," in IEEE Access, vol. 11, pp. 107972-107981, 2023, doi: 10.1109/ACCESS.2023.3313508

7. A. Rehman, M. Kashif, I. Abunadi and N. Ayesha, "Lung Cancer Detection and Classification from Chest CT Scans Using Machine Learning Techniques", 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA), pp. 101-104, 2021.
8. R. D. Karthikeyan, R. G. V. V. G. B. C and K. M, "A Review of Lung Cancer Detection using Image Processing", 2021 Smart Technologies. Communication and Robotics (STCR), pp. 1-4, 2021.
9. A. Jemal, R. Siegel, J. Xu and E. Ward, "Cancer statistics 2010", CA. Cancer J. Clin., vol. 60, no. 5, pp. 277-300.
10. FR Hirsch, W A Franklin, Af Gazdar and P A.j Bunn, "Early detection of lung cancer: clinical perspectives of recent advances in biology and radiology", Clin Cancer Res, vol. 7, pp. 5-22, 2001.