

Enhanced Lung Cancer Detection Using Deep Learning Ensemble Approach

*A Project Report submitted in the partial fulfillment
of the Requirements for the award of the degree*

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING

Submitted by

Ainavolu Manvitha (21471A0572)

Dokku Naga Revathi (21471A0581)

Kasula Prathima (21471A0595)

Under the esteemed guidance of

Dr. Sireesha Moturi, B. Tech, M. Tech, Ph.D.,

Associate Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET
(AUTONOMOUS)**

Accredited by NAAC with A+ Grade and NBA under Tire -1 NIRF rank in the band of
201-300 and an ISO 9001:2015 Certified

Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada
KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, NARASARAOPET- 522601
2024-2025

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)
**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING**



CERTIFICATE

This is to certify that the project that is entitled with the name “**Enhanced Lung Cancer Detection Using Deep Learning Ensemble Approach**” is a bonafide work done by the team **Ainavolu Manvitha (21471A0572)**, **Dokku Naga Revathi (21471A0581)**, **Kasula Prathima (21471A0595)** in partialfulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2024-2025.

PROJECT GUIDE

Dr. Sireesha Moturi, B.Tech., M.Tech., Ph.D.,
Associate Professor

PROJECT CO-ORDINATOR

Dr. Sireesha Moturi, B.Tech., M.Tech., Ph.D.,
Associate Professor

HEAD OF THE DEPARTMENT

Dr. S. N. Tirumala Rao, B.Tech., M.Tech., Ph.D.,
Professor & HOD

EXTERNAL EXAMINER

DECLARATION

We declare that this project work titled **“Enhanced Lung Cancer Detection Using Deep Learning Ensemble Approach”** is composed by ourselves that the work contained here is our own except where explicitly stated otherwise in the text and that this work has not been submitted for anyother degree or professional qualification except as specified.

By

Ainavolu Manvitha (21471A0572)

Dokku Naga Revathi (21471A0581)

Kasula Prathima (21471A0595)

ACKNOWLEDGEMENT

We wish to express our thanks to various personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman Sri **M. V.Koteswara Rao**, B.Sc., who took keen interest in us in every effort throughout this course. We owe out sincere gratitude to our beloved principal **Dr. S. Venkateswarlu**, Ph.D., for showing his kind attention and valuable guidance throughout the course.

We express our deep felt gratitude towards **Dr. S. N. Tirumala Rao**, B.Tech., M.Tech., Ph.D., HOD of CSE department and also to our guide **Dr. Sireesha Moturi**, B.Tech., M.Tech., Ph.D., of CSE department whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks towards **Dr. Sireesha Moturi**, B.Tech., M.Tech., Ph.D., Associate professor & Project coordinator of the project for extending her encouragement. Their profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

We extend our sincere thanks to all other teaching and non-teaching staff to department for their cooperation and encouragement during our B.Tech degree.

We have no words to acknowledge the warm affection, constant inspiration and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends and those who involved in giving valuable suggestions had clarifying out doubts which had really helped us in successfully completing our project.

By

Ainavolu Manvitha (21471A0572)

Dokku Naga Revathi (21471A0581)

Kasula Prathima (21471A0595)



INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community,

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems

**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING**

VISION OF THE DEPARTMENT

To become a centre of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertise in modern software tools and technologies to cater to the real time requirements of the Industry.

M3: Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

Program Educational Objectives (PEO's)

The graduates of the programme are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems related to academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

Program Outcomes

- 1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- 3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.
- 4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- 5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- 6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

- 7. Environment and sustainability:** Understand the impact of the professional solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- 8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- 9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- 10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- 11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- 12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Project Course Outcomes (CO'S):

CO421.1: Analyse the System of Examinations and identify the problem.

CO421.2: Identify and classify the requirements.

CO421.3: Review the Related Literature

CO421.4: Design and Modularize the project

CO421.5: Construct, Integrate, Test and Implement the Project.

CO421.6: Prepare the project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1		✓											✓		
C421.2	✓		✓		✓								✓		
C421.3				✓		✓	✓	✓					✓		
C421.4			✓			✓	✓	✓					✓	✓	
C421.5					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C421.6									✓	✓	✓		✓	✓	

Course Outcomes – Program Outcome correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1	2	3											2		
C421.2			2		3								2		
C421.3				2		2	3	3					2		
C421.4			2			1	1	2					3	2	
C421.5					3	3	3	2	3	2	2	1	3	2	1
C421.6									3	2	1		2	3	

Note: The values in the above table represent the level of correlation between CO's and PO's:

1. Low level
2. Medium level
3. High level

Project mapping with various courses of Curriculum with Attained PO's:

Name of the course from which principles are applied in this project	Description of the device	Attained PO
C2204.2, C22L3.2	Gathering the requirements and defining the problem, plan to develop an Ensemble 2D CNN model for recognizing a Lung Nodule either cancerous or not	PO1, PO3
CC421.1, C2204.3, C22L3.2	Each and every requirement is critically analyzed, the process model is identified	PO2, PO3
CC421.2, C2204.2, C22L3.3	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9
CC421.3, C2204.3, C22L3.2	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5
CC421.4, C2204.4, C22L3.2	Documentation is done by all our four members in the form of a group	PO10
CC421.5, C2204.2, C22L3.3	Each and every phase of the work in group is presented periodically	PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Implementation is done and the project will be handled in the medical and healthcare industries to detect lung cancer effectively.	PO4, PO7
C32SC4.3	The physical design includes a web access application which accepts an input image and detects whether the input Lung CT scan image is cancerous or not.	PO5, PO6

ABSTRACT

In the era of emerging technologies, Deep Learning has become a transformative solution for addressing real-world challenges, particularly in the medical domain. One critical application is the early detection of lung cancer, a leading cause of mortality worldwide. Accurately classifying lung nodules as cancerous or non-cancerous remains a significant challenge in medical imaging. While various studies have explored Deep Learning-based approaches for lung nodule detection, leveraging an ensemble of models can enhance predictive accuracy and robustness. In this study, we propose a novel ensemble-based Deep Learning methodology that integrates three Convolutional Neural Network (CNN) models for improved lung cancer detection. Instead of relying on a single model, our approach combines multiple CNN architectures to extract diverse features from Computed Tomography (CT) scan images, leading to more reliable classification results. We utilize the publicly available LUNA16 dataset from Grand Challenge, which comprises CT scan images along with annotated nodule information. Experimental results demonstrate that our ensemble 2D CNN model achieves an accuracy of 96%, outperforming baseline methodologies and demonstrating its potential for early and precise lung cancer diagnosis.

INDEX

S.NO.	CONTENT	PAGE NO
1.	Introduction	01
2.	Literature Survey	
	2.1. Related Work	04
	2.2 Applications of Deep Learning	06
3.	Existing System	07
4.	Proposed Methodology	08
5.	System Requirements	
	5.1. Software Requirements	09
	5.2. Hardware Requirements	09
6.	System Analysis	
	6.1. Scope of the Project	10
	6.2. Analysis	11
	6.3 Data Collection	12
	6.4. Data Pre-processing	13
	6.5. Splitting the Dataset	14
	6.6. Model Building	14
	6.7. Classification	18
	6.8. Evaluation Metrics	18
7.	Design	20
8.	Implementation	21
9.	Result Analysis	45
10.	Test Cases	51
11.	User Interface	53
12.	Conclusion	55
13.	Future Scope	56
14.	References	57

LIST OF FIGURES

S.NO.	LIST OF FIGURES	PAGE NO
1.	Fig 1.1: Lung Cancer Survival Rate Over Time	02
2.	Fig 4.1: Flowchart of Proposed Methodology	08
3.	Fig 6.1: CT Scan images before Preprocessing	12
4.	Fig 6.2: CT Scan images after Preprocessing	13
5.	Fig 7.1: Design Overview	20
6.	Fig 9.1: Confusion Matrix of 2D CNN1	45
7.	Fig 9.2: Classification Report of 2D CNN1	45
8.	Fig 9.3: Accuracy and Loss Curves of 2D CNN1	46
9.	Fig 9.4: Confusion Matrix of 2D CNN2	46
10.	Fig 9.5: Classification report of 2D CNN2	47
11.	Fig 9.6: Accuracy and Loss Curves of 2D CNN1	47
12.	Fig 9.7: Confusion Matrix for 2D CNN3	48
13.	Fig 9.8: Classification Report of 2D CNN3	48
14.	Fig 9.9: Accuracy and Loss Curves of 2D CNN1	49
15.	Fig 9.10: Confusion Matrix for Ensemble 2D CNN	49
16.	Fig 9.11: Classification report of Ensemble 2D CNN	50
17.	Fig 9.12: Accuracy and Loss Curves of 2D CNN1	50
18.	Fig 10.1 Detected Cancer	51
19.	Fig 10.2 Detected Non-Cancer	51
20.	Fig 10.3 Detecting Invalid Input Image	52
21.	Fig 11.1 Home Screen	53
22.	Fig 11.2 Symptoms screen	53
23.	Fig 11.3 Output screen	54

LIST OF TABLES

S.NO	LIST OF TABLES	PAGENO
1.	Table 6.1 Dataset Description	11
2.	Table 6.2: Model Architecture for 2D CNN1	14
3.	Table 6.3: Model Architecture for 2D CNN2	16
4.	Table 6.4 Model Architecture for 2D CNN3	17

1.INTRODUCTION

1.1 Introduction

Lung Cancer has become the most frequent disease in these days. Many people in the world are suffering from this problem for not identifying this disease early. The development of technology has also increased many solutions in the medical industry. However, no solution detects lung cancer through the individual symptoms [1].

Mostly detecting through medical images is a quite difficult task to identify whether the person is prone to cancer or not. The United States has registered the greatest number of cases worldwide. In the year 2018, the GLOBOCAN estimated that nearly 2 million new cases were registered due to lung cancer [2]. Traditional diagnostic methods, such as biopsies and radiographic imaging, although effective, are time-consuming, invasive, and prone to human error. Medical professionals often rely on computed tomography (CT) scans for detecting lung abnormalities, including pulmonary nodules. While CT scans provide high-resolution images, accurately identifying and classifying nodules from these scans is challenging even for experienced radiologists, as nodules can vary widely in size, shape, and texture [3].

Advances in technology, particularly in the fields of artificial intelligence (AI) and Deep Learning, have revolutionized the approach to medical imaging. AI-based solutions offer non-invasive, fast, and accurate diagnostic capabilities, aiding radiologists in decision-making and reducing the workload [4]. Among these solutions, Deep Learning has emerged as a game-changer due to its ability to process and analyze large datasets, identify patterns, and predict outcomes with high precision.

In the context of lung cancer detection, convolutional neural networks (CNNs), a specialized class of Deep Learning models, have demonstrated exceptional performance in image analysis tasks. CNNs mimic the way humans perceive visual data, automatically learning to extract features from raw images without the need for manual feature engineering. This capability makes CNNs particularly suited for the analysis of CT scans, enabling them to differentiate between malignant and benign nodules with remarkable accuracy.

One of the leading technologies is Deep Learning which contains a huge number of libraries and methods to classify a task. To identify if a person is Cancerous or Non Cancerous the way is to analyze CT Scans. Deep Learning has given a way to identify pulmonary nodules in CT scans whether Cancerous or not.

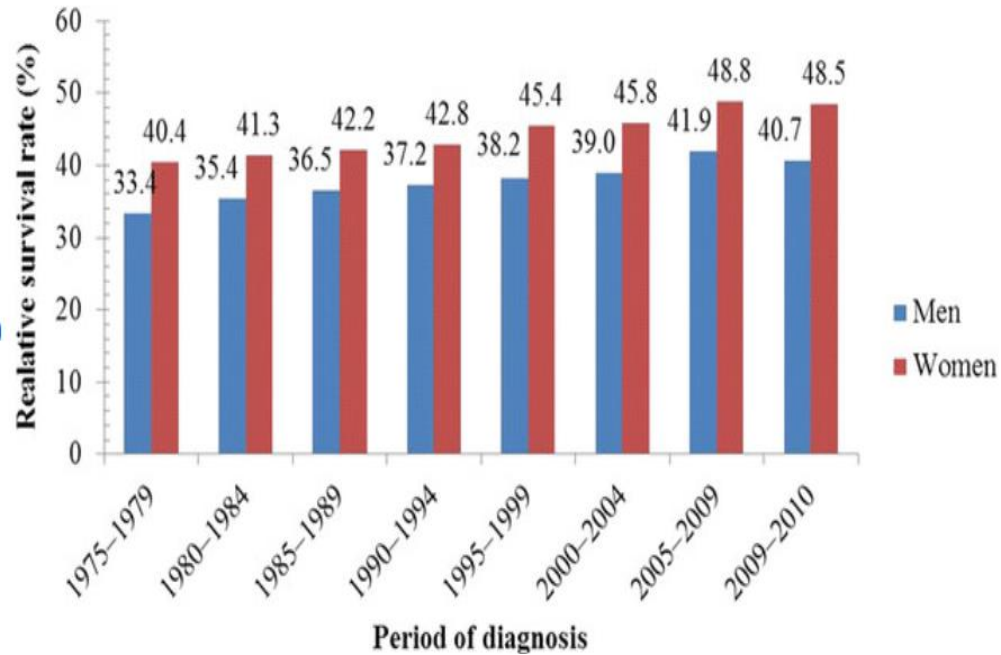


Fig 1.1: Lung Cancer Survival Rate Over Time

The figure 1.1 shows the relative survival rates of lung cancer patients over different diagnosis periods. Survival rates have improved over time, with women consistently having higher rates than men.

The state of the nodule in the CT scan can be malignant and benign. To classify them correctly from the input CT scan we should use Neural Networks. They take images as input, learn the patterns in the image, and can find the nodule. Another important task in classifying these images is handling three-dimensional images.

Medical images are visualized in 3D view to analyze clearly. So, to handle these types of images, efficiency in identifying them is needed. DL approaches introduce Neural networks, which learn the patterns of images in all dimensions. Deeper CNNs like GoogleNet, AlexNet, and ResNet became advanced in analyzing the input patterns.

Pulmonary nodules are small, round growths in the lung that can be benign or malignant. Accurate classification of these nodules is essential for determining the appropriate course of treatment. The complexity of CT scan images, combined with the

variability in nodule appearance, makes this task particularly challenging for traditional diagnostic methods. Deep Learning models, particularly convolutional neural networks (CNNs), have demonstrated their ability to extract meaningful features from medical images, providing a robust solution for this problem [5].

From this we have chosen CNNs as it has multi view architecture in processing input images. However, there are some disadvantages to this. Only certain features of lung nodules are identified, which leads to incorrect prognosis and incorrect classification. In this study, we have improved the CNN to classify the lung Nodules.

Our research is based on the LUNA16 dataset, a widely recognized benchmark dataset in the domain of lung cancer detection. The dataset provides a rich repository of annotated CT scans, enabling the development and validation of Deep Learning models. Through our ensemble approach, we seek to overcome the limitations of individual CNN models and achieve higher diagnostic accuracy, ultimately contributing to the early detection of lung cancer

2.LITERATURE SURVEY

2.1 Related Work

Out of studies conducted, an automatic detection algorithm using Deep Learning can reduce overlooked lung cancers on chest radiographs without a proportional increase in follow-up chest CT examinations. The study covers the Deep Learning-based Automatic Detection Algorithm for Reducing Overlooked Lung Cancers on Chest Radiographs [6].

A promising outcome has been achieved in the use of 2D Convolutional Neural Networks combined with Taguchi parametric optimization in lung cancer detection from CT images. The approach is followed in trying to improve the accuracy of a CNN model that will classify lung cancer [7]. This optimization increased the accuracy from 91.97% to 98.83% proving that tuning parameters can improve the performance of the models.

One of the research studies proposes the ensemble model that combines the outputs of different deep-learning architectures: CNNs and RNNs. The strengths of several models are combined using the ensemble approach in order to enhance robustness and generalizability [8]. In this respect, this model achieved an accuracy of 77.6% for invasive adenocarcinoma 2 risk stratification.

Optimal Deep Neural Networks and Convolutional Neural Networks that went through this research have demonstrated very high accuracy, sensitivity, and specificity in the classification of lung nodules as malignant or benign using CT images. It attained an accuracy of 95.60% and a log loss of 0.387732 [9].

Hybrid bio-inspired algorithm and convolutional neural network for automatic lung tumor detection. The proposed hybrid WOA_APSO algorithm with a Convolutional Neural Network showed a promising result in the effective detection of lung tumors with good accuracy, sensitivity, and specificity by achieving an accuracy of 97.18% [10], a sensitivity of 97%, and a specificity of about 98.66.

This study brings into view that computer-aided diagnosis using computed tomography (CT) scan images increases the accuracy of detection in lung cancer, but most of the current methods need further development to achieve 100% accuracy.

Deep Learning techniques, especially convolutional neural networks (CNNs), have achieved great results in classifying subtypes of lung cancers such as adenocarcinoma (LUAD) and squamous cell carcinoma (LUSC) from histopathology images with results achieving an AUC of 0.97 comparable with pathologists [11].

The greater efficacy of clustering techniques and integration of Deep Learning models in lung cancer detection from CT images has risen the diagnostic accuracy to a great extent. While the classification error was 38% [12], the accuracy rate of the system was 98.42. Some reviews say that AlexNet, when combined with various classifiers, offers accuracy as high in the detection of lung cancer, while some say that VGG-16, ResNet, and Inception provide state-of-the-art performance, further achieving as high as 100 % accuracy.

These studies concluded that Deep Learning models, especially CNNs, including Inception V3, report a high value of accuracy, precision, recall, and specificity concerning the detection of lung cancer using CT images.

2.2 Applications of Deep Learning

1. Image Recognition and Classification
2. Natural Language Processing (NLP)
3. Speech Recognition and Synthesis
4. HealthCare
5. Finance
6. Autonomous Systems
7. Recommendation Systems
8. Robotics
- 9 Image Captioning
10. Education and Research
11. Self Driving Cars
12. Natural Language Processing
13. Climate and Environment Science
14. Fraud Detection
15. Detecting Developmental Delay in Children
16. Colourisation of Black and White images
17. Manufacture and Industry
18. Automatic Machine Translation
19. Generative Modeling

3.EXISTING SYSTEM

Manual Diagnosis by Radiologists

Traditional diagnosis relies on radiologists analyzing chest X-rays or CT scans to identify abnormalities in the lungs. This process is time-consuming, prone to human error, and highly dependent on the expertise of the radiologist. Subtle nodules or small tumors are often missed, especially in early stages.

Conventional Machine Learning Approaches

Early AI-based methods used traditional machine learning algorithms (e.g., support vector machines, decision trees) to classify lung nodules. These systems relied heavily on handcrafted feature extraction, which limited their ability to generalize across different datasets and imaging variations.

Lack of Early Detection Tools

Many existing systems are geared toward diagnosing advanced stages of lung cancer when symptoms become apparent. There is a lack of tools capable of identifying early-stage cancer, which is critical for improving survival rates.

Data Imbalance Issues

Traditional systems struggle with imbalanced datasets where malignant cases are fewer compared to benign ones, leading to biased predictions and reduced sensitivity to cancerous cases.

Image Processing Techniques

Basic image processing methods like thresholding, edge detection, and segmentation have been used to identify regions of interest in lung images. These techniques often struggle with noise, variability in nodule shapes, and overlapping structures, reducing their accuracy.

4. PROPOSED METHODOLOGY

Earlier studies have problems with ensemble learning techniques. In this work, we propose an ensemble of three different 2D CNNs for the detection of lung cancer from CT images. The proposed methodology shown in the figure 4.1 involves the below steps:

- **Data Collection:** The dataset used for lung cancer detection consists of CT scan images, sourced from publicly available datasets like LUNA16 and Kaggle.
- **Data Pre-processing:** The data is preprocessed such that the meta data is converted into images and images are cropped around the lung nodules.
- **Splitting the dataset :** The dataset is divided into training, test sets, ensuring a balanced distribution of cancerous and non-cancerous cases.
- **Building the Model Architecture:** Three different 2D CNN models are designed with varied depths, kernel sizes, and filter configurations to extract diverse features.
- **Training the Model train dataset :** Each CNN model is trained independently using the training dataset with appropriate hyperparameters like learning rate, batch size, and number of epochs.
- **Testing the Model with the test dataset :** Soft voting is applied to combine predictions from the three models, enhancing classification accuracy.

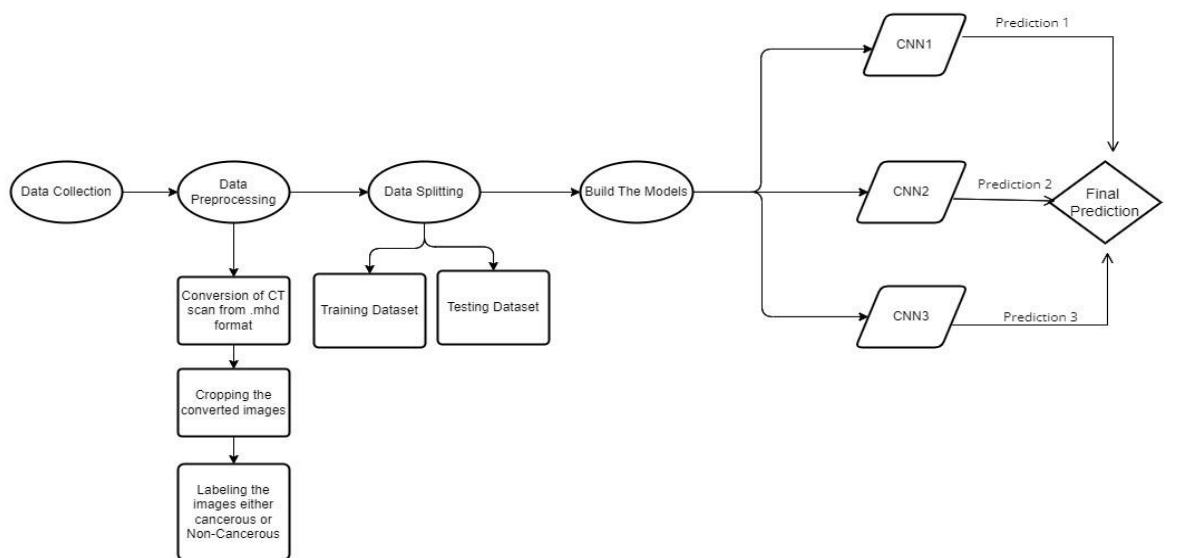


Fig 4.1: Flowchart of Proposed Methodology.

5. SYSTEM REQUIREMENTS

5.1. Hardware Requirements:

- System Type : intel®core™i3-7500UCPU@2.40gh
- Cache memory : 4MB(Megabyte)
- RAM : 8GB (gigabyte)
- Hard Disk : 4GB

5.2. Software Requirements:

- Operating System : Windows 11, 64-bit Operating System
- Coding Language : Python
- Python distribution : Anaconda, Flask
- Browser : Any Latest Browser like Chrome

6. SYSTEM ANALYSIS

6.1 Scope of the project

Medical Relevance: This project focuses on lung cancer detection, a critical medical challenge due to the high mortality rate associated with late-stage diagnosis. Early detection of lung cancer can significantly improve patient survival rates, making this project highly impactful for healthcare.

Deep Learning Approach: The project employs a Deep Learning-based ensemble of 2D Convolutional Neural Networks (CNNs) to analyze medical images. This approach enhances the detection process, leveraging multiple CNN models to improve the accuracy and robustness of predictions.

Dataset Utilization: The project utilizes two prominent datasets, LUNA16 and a supplementary Kaggle dataset, which contain annotated Computed Tomography (CT) scan images of lung nodules. These datasets provide a diverse and comprehensive collection of images to train and evaluate the ensemble model.

Medical Image Processing: The scope of this project includes extensive image preprocessing steps. These steps involve converting .mhd images to JPEG format, voxel coordinate transformation, image normalization, and data augmentation to address class imbalances.

Ensemble Model Integration: The ensemble approach integrates multiple CNN models to enhance prediction accuracy. By combining the predictions of several models, the system achieves higher robustness and reduces false positives and false negatives in lung cancer classification.

Impact on Healthcare: The project's primary goal is to develop a reliable and accurate system for early lung cancer detection. This can contribute to early intervention, reduce the burden on healthcare systems, and ultimately save lives by enabling timely treatment.

6.2 Analysis

Data Preparation: The analysis begins with the preprocessing of the LUNA16 and Kaggle datasets. The raw medical CT images are converted into a 2D format suitable for CNN analysis. This step also includes resizing images to a uniform 50x50 pixel size and normalizing pixel intensities to enhance model performance.

Annotation and Labelling: Lung nodule annotations in the LUNA16 dataset are converted into voxel coordinates for precise localization within the CT images. This facilitates the accurate extraction of regions of interest (ROIs) required for training and testing.

Ensemble Model Architecture: The proposed ensemble consists of three 2D CNN models with distinct architectures. Each CNN model employs different configurations of convolutional layers, kernel sizes, pooling layers, and activation functions. The ensemble aggregates the predictions from these models to achieve higher accuracy.

Model Training and Evaluation: The models are trained using the training set, and hyperparameters such as learning rate, batch size, and dropout rates are fine-tuned to optimize performance. Cross-validation techniques are used to assess the generalizability of the model on unseen data.

Performance Metrics: The project evaluates model performance using metrics such as accuracy, precision, recall, and F1-score. The ensemble model aims to maximize accuracy and reduce false negatives, which are critical in lung cancer detection.

Comparative Analysis: The analysis includes a comparative study of different CNN model architectures. Key performance metrics of individual models and the final ensemble model are compared to highlight the effectiveness of the ensemble approach.

6.3. Data Collection

Lung Nodule Analysis Dataset 2016 (LUNA16) is an important source for investigators involved in lung nodule detection. The dataset comes from the bigger LIDC/IDRI database that includes 1,018 CT scans each annotated by four experienced radiologists. However, within this database, nodules are divided into three categories: nodules greater than or equal to 3 mm, nodules less than 3 mm, and non-nodules.[4] Last, it comprises annotations indicating the centroid locations and the diameters of detected pulmonary nodules for all of its chest CT images counting to 888.

To keep results consistent, users should use these subsets in 10-fold cross-validation. Each of these contains CT images along with several key files such as annotations.csv (reference annotations), sampleSubmission.csv (an example submission format), candidates.csv (basic candidate sites), candidates_V2.csv (extended candidate sites), evaluation scripts, lung segmentation data, additional_annotations.csv. The Dataset can be accessed from LUNA16(Lung Nodule Analysis 2016) Grand Challenge, it has become a widely used source for research, especially on pulmonary nodule detection and related fields.

Table 6.1 Dataset Description

Category	Count	Cancer	Non-Cancer	Pixel size	format
Candidates.csv	551065	1359	549714	-	-
Annotations.csv	1186	-	-	-	-
CT Scan images	888	-	-	512x512	.mhd

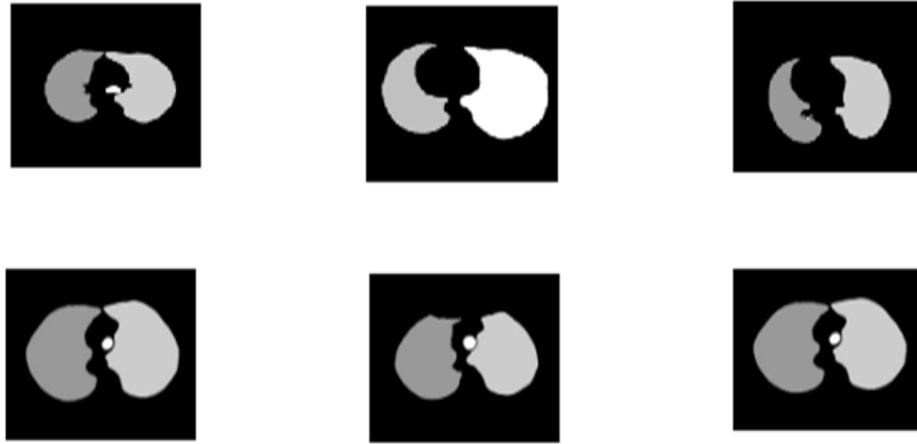


Fig 6.1: CT Scan images before Pre-processing

6.4. Data Preprocessing

The dataset utilized in this paper is the LUNA16 dataset sourced from Grand Challenge 2016. This dataset contains data about lung cancer with the CT scan images in .mhd format.

- .mhd files contain image resolution, pixel spacing, and data type metadata. This information correctly interprets the associated .raw file, which contains the raw pixel data.
- This is done with libraries such as SimpleITK. The images after converting from .mhd to JPEG format are visualized, and Figure shows the images .
- Once the image is reconstructed, the image can be saved in .JPEG format using a standard image processing library PIL (Python Imaging Library). After converting the images from MHD to JPEG format and applying normalization.
- The images in JPEG format shown in Fig 6.3.2 are cropped in the region based on the coordinates given in the annotations.csv file, which gives information about each CT scan. Figure 6.3.3 shows the preprocessed images.

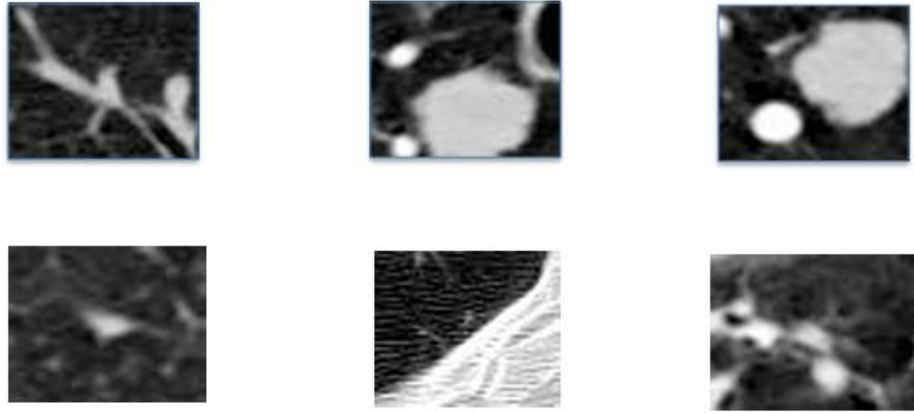


Fig 6.2: CT Scan images after pre-processing

6.5. Splitting the Dataset:

The LUNA16 dataset was split into training and testing sets for model training and evaluation. The split was performed based on the `class` column in the `candidates.csv` file, which indicates whether a candidate is cancerous (1) or non-cancerous (0).

Instead of organizing the data into separate subfolders for "Cancerous" and "Non-Cancerous," the dataset was directly divided into two folders:

- **Train:** Used for training the model to learn from the data.
- **Test:** Used to evaluate the model's accuracy and prediction capabilities.

This structure simplifies the workflow by grouping all data into two distinct sets, ensuring efficient model development and validation after preprocessing steps.

6.6 Model Building and Training :

Once the splitting is done, the next important step is to build the model architecture and train the model. The algorithm we have chosen is Ensemble 2D CNN. For this Ensemble technique, we trained three 2D CNN models. with different layers and combine the models' predictions to get the ensemble results. The model architectures of the three models are described below:

6.6.1 Ensemble 2D CNN:

For the first model, a convolutional neural network architecture consisting of three convolutional blocks was adopted, each followed by a batch normalization layer, a max pooling layer, and a dropout layer. The convolutional layers use 32, 64, and 128 size filters respectively, each with a kernel size of 3x3, and the ReLU activation function. This is so that the 'he_uniform' initializer sets weights to small random values optimized for ReLU activation. A dense layer with softmax activation serves to output class probabilities. It has employed focal loss in training for handling the class imbalance problem, and for optimization, it uses the Adam optimizer.

Table 6.2: Model Architecture for 2D CNN1

Layer (Type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 50, 50, 32)	320
batch_normalization_7	(None, 50, 50, 32)	128
conv2d_7 (Conv2D)	(None, 50, 50, 32)	9,248
batch_normalization_8	(None, 50, 50, 32)	128
max_pooling2d_3	(None, 25, 25, 32)	0
dropout_4 (Dropout)	(None, 25, 25, 32)	0
conv2d_8 (Conv2D)	(None, 25, 25, 64)	18,496
batch_normalization_9	(None, 25, 25, 64)	256
conv2d_9 (Conv2D)	(None, 25, 25, 64)	36,928
batch_normalization_10	(None, 25, 25, 64)	256
max_pooling2d_4	(None, 12, 12, 64)	0
dropout_5 (Dropout)	(None, 12, 12, 64)	0
conv2d_10 (Conv2D)	(None, 12, 12, 128)	73,856
batch_normalization_11	(None, 12, 12, 128)	512
conv2d_11 (Conv2D)	(None, 12, 12, 128)	147,584
batch_normalization_12	(None, 12, 12, 128)	512
max_pooling2d_	(None, 6, 6, 128)	0
dropout_6 (Dropout)	(None, 6, 6, 128)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_2 (Dense)	(None, 128)	589,952
batch_normalization_13	(None, 128)	512
dropout_7 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 2)	258

The second model deepens the architecture by the introduction of a fourth convolution block, scaling the depth of features to be extracted from 64 to 512 filters. Each convolution block introduces a batch normalization layer, and the next layers are the max-pooling layer and a dropout layer. The dropout rate increases progressively from 0.2 to 0.5. The activation function remains ReLU for hidden layers, with the initializer as 'he_uniform' for optimizing weight distribution. The fully connected ending layer is of dimension 256 units, the activation ReLU. The softmax function will be used for the final output layer to give class probabilities. In this model, Focal loss is similar to the first CNN model.

The third model is a bit more complex, which will define a CNN architecture with three convolutional blocks followed by batch normalization, ReLU, and dropout layers, respectively. The first two convolution blocks will contain 32 and 64 filters, each with kernel sizes 3×3 and 5×5 , respectively, while the third block has 128 with kernel size 3×3 . After that, the model is terminated by a flattening layer, which is followed by a fully connected layer of units 84, activated by ReLU. Lastly, for output, softmax activation has been used to handle binary classification problems. Each model produced a probability vector after training over the probability of the input image being in either of the two classes.

Table 6.3: Model Architecture of 2D CNN2

Layer (Type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 50, 50, 64)	640
batch_normalization_9	(None, 50, 50, 64)	256
conv2d_9 (Conv2D)	(None, 50, 50, 64)	36,928
batch_normalization_10	(None, 50, 50, 64)	256
max_pooling2d_5	(None, 25, 25, 64)	0
dropout_5 (Dropout)	(None, 25, 25, 64)	0
conv2d_10 (Conv2D)	(None, 25, 25, 128)	73,856
batch_normalization_12	(None, 25, 25, 128)	512
conv2d_11 (Conv2D)	(None, 25, 25, 128)	147,584
batch_normalization_12	(None, 25, 25, 128)	512
max_pooling2d_5	(None, 12, 12, 128)	0
dropout_6 (Dropout)	(None, 12, 12, 128)	0
conv2d_12 (Conv2D)	(None, 12, 12, 256)	295,168
batch_normalization_13	(None, 12, 12, 256)	1,024
conv2d_13 (Conv2D)	(None, 12, 12, 256)	590,080
batch_normalization_14	(None, 12, 12, 256)	1,024
max_pooling2d_6	(None, 6, 6, 256)	0
dropout_7 (Dropout)	(None, 6, 6, 256)	0
conv2d_14 (Conv2D)	(None, 6, 6, 512)	1,180,160
batch_normalization_15	(None, 6, 6, 512)	2,048
conv2d_15 (Conv2D)	(None, 6, 6, 512)	2,359,808
batch_normalization_16	(None, 6, 6, 512)	2,048
max_pooling2d_7	(None, 3, 3, 512)	0
dropout_8 (Dropout)	(None, 3, 3, 512)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_2 (Dense)	(None, 256)	1,179,904
batch_normalization_17	(None, 256)	1,024
dropout_9 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 2)	514

Table 6.4: Model Architecture of CNN3

Layer (Type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 50, 50, 32)	320
batch_normalization_3	(None, 50, 50, 32)	128
max_pooling2d_3	(None, 25, 25, 32)	0
conv2d_4 (Conv2D)	(None, 25, 25, 64)	18,406
batch_normalization_4	(None, 25, 25, 64)	256
max_pooling2d_4	(None, 12, 12, 64)	0
conv2d_5 (Conv2D)	(None, 12, 12, 128)	73,856
batch_normalization_5	(None, 12, 12, 128)	512
max_pooling2d_5	(None, 6, 6, 128)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_2 (Dense)	(None, 128)	589,952
dropout_1 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 64)	8,256
dropout_2 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 1)	65

6.7. Classification:

To analyze the model we obtained Accuracy, Precision, and recall are compared as follows:

6.7.1 Ensemble 2D CNN model:

The above model architectures were compiled and trained on the dataset. The results and analysis of the models were described below:

- 2D CNN1:** The model's overall effectiveness was quite high, at an accuracy of 95%, but with good precision on non-cancerous cases classification at 0.98. The model is trained for 70 epochs. In the case of the cancerous cases, this recall was higher at 0.91, meaning that more true positives were likely to be caught by the

model, but at the expense of a higher rate of false positives. The F1-score for the cancerous class showed an instance balance between precision and recall at 0.82. The results and ROC curves of CNN1 are shown below

- **2D CNN2:** The CNN2 model performed very well in differentiating cancerous from non-cancerous cases, reflected by a testing accuracy of 94.91%. The high precision and recall values for both classes demonstrate that the model is good in predicting cancerous cases. However, it tends to have a bit more false negatives, 45 cases, compared with false positives, 27 cases.
- **2D CNN3:** Subsequently, the model was used in testing data to estimate the result of the CNN in line with the results obtained from the study. The first model of CNN in iteration shows us an accuracy of 91%. According to the AUC accuracy values would be considered as excellent. The model was trained with 50 epochs.
- **Ensemble 2D CNN:** The probabilities of the three models are combined to create the ensemble model. As the models were compiled with different input layers and activation functions the performances will be combined. With accuracies averaging around that mark for three different models, After combining the three models we obtained the accuracy of 96%. This is because an ensemble has the added advantage of not overfitting and learning varied patterns from the dataset, which yields strong and robust predictions.

6.8 Evaluation Metrics:

To assess the performance of our classification model, we employ various evaluation metrics that provide insights into its effectiveness in identifying diabetic retinopathy. The key metrics used include accuracy, precision, recall, F1 score, and time complexity.

- **Accuracy:** Accuracy measures the proportion of correctly classified instances out of the total instances. It is a fundamental metric for assessing model performance.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

where TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives.

- **Precision:** Precision quantifies how many of the predicted positive instances are actually positive. It is crucial when false positives need to be minimized.

$$\text{Precision} = \frac{TP}{TP+FP}$$

- **Recall:** Recall measures the model's ability to identify actual positive cases. A high recall indicates that the model captures the most relevant instances.

$$\text{Recall} = \frac{TP}{TP+FN}$$

- **F1Score:** The F1 Score is the harmonic mean of precision and recall, providing a balanced measure when both are important.

$$\text{F1 Score} = \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Time Complexity:** Time complexity refers to the computational cost of running the model in terms of processing time. It depends on factors such as dataset size, number of features, and algorithm efficiency.

7. DESIGN

The process in the figure 7.1 begins with data collection, followed by preprocessing to prepare the data for analysis, including techniques like augmentation to address variability. The dataset is then split into training, validation, and test sets. Three distinct 2D CNN models with varied architectures are built and trained independently to extract diverse features from the data. After training, the models are saved, and their predictions are ensembled using the Soft Voting method to enhance overall performance. Finally, the ensemble model classifies inputs into cancerous or non-cancerous categories, with the results displayed in a report.

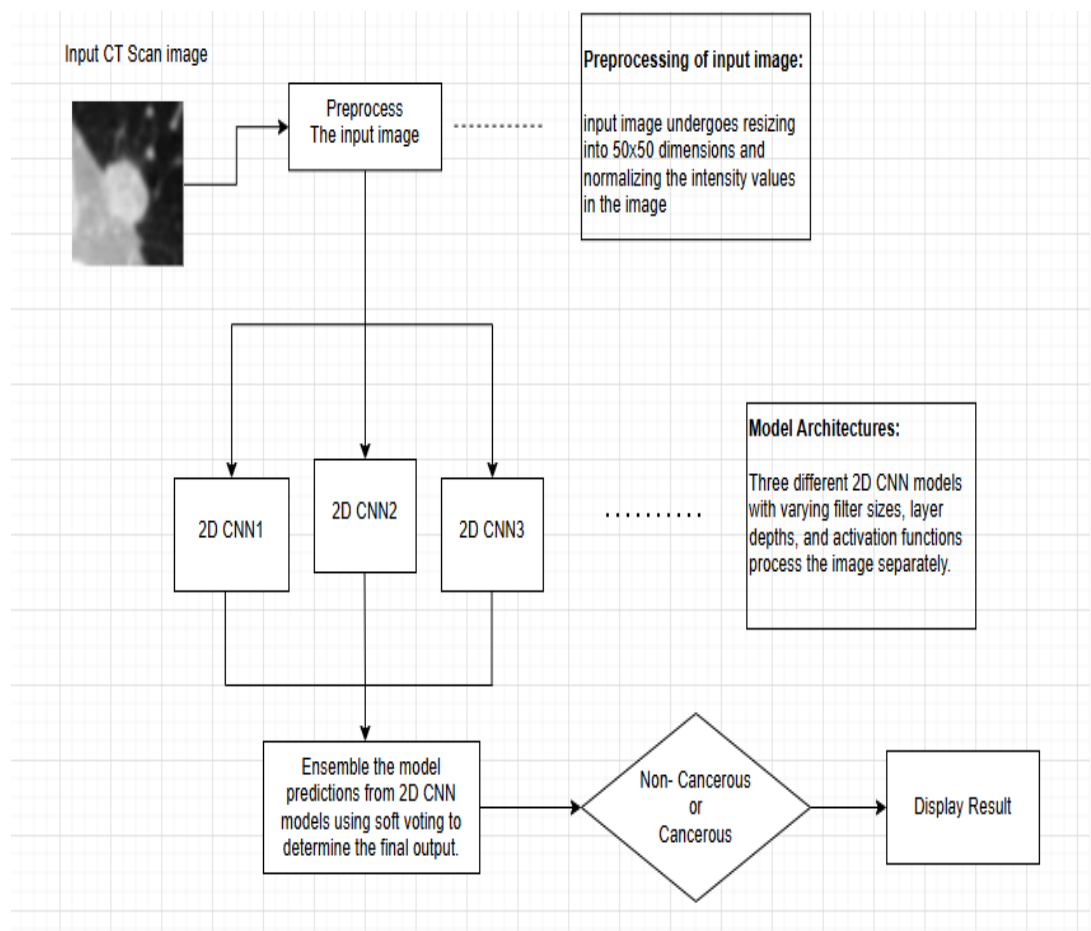


Fig 7.1: Design Overview

8. IMPLEMENTATION

Data Collection:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import cv2
import random

annotations=pd.read_csv('/content/drive/MyDrive/Dataset/annotations.csv')
candidates=pd.read_csv('/content/drive/MyDrive/Dataset/candidates.csv')
annotations.head()

positives = candidates[candidates['class']==1].index
negatives = candidates[candidates['class']==0].index
np.random.seed(42)
negIndexes = np.random.choice(negatives, len(positives)*5, replace = False)
candidatesDf = candidates.iloc[list(positives)+list(negIndexes)]
```

Data Pre-processing

```
from sklearn.model_selection import train_test_split
X = candidatesDf.iloc[:, :-1]
y = candidatesDf.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state
= 42)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size = 0.20,
random_state = 42)
X_train.to_pickle('/content/drive/MyDrive/Dataset/Preprocessed_data/traindata')
X_test.to_pickle('/content/drive/MyDrive/Dataset/Preprocessed_data/testdata')
X_val.to_pickle('/content/drive/MyDrive/Dataset/Preprocessed_data/valdata')
import pandas as pd
X_train=pd.read_pickle('/content/drive/MyDrive/Dataset/Preprocessed_data/traindata')
X_test = pd.read_pickle('/content/drive/MyDrive/Dataset/Preprocessed_data/testdata')
```

```

X_val = pd.read_pickle('/content/drive/MyDrive/Dataset/Preprocessed_data/valdata')
def normalizePlanes(npzarray):
    maxHU = 400.
    minHU = -1000.
    npzarray = (npzarray - minHU) / (maxHU - minHU)
    npzarray[npzarray>1] = 1.
    npzarray[npzarray<0] = 0.
    return npzarray
tempDf = X_train[y_train == 1]
tempDf = tempDf.set_index(X_train[y_train == 1].index + 1000000)
X_train_new = pd.concat([X_train, tempDf])
tempDf = tempDf.set_index(X_train[y_train == 1].index + 2000000)
X_train_new = pd.concat([X_train_new, tempDf])
ytemp = y_train.reindex(X_train[y_train == 1].index + 1000000)
ytemp.loc[:] = 1
y_train_new = pd.concat([y_train, ytemp])
ytemp = y_train.reindex(X_train[y_train == 1].index + 2000000)
ytemp.loc[:] = 1
y_train_new = pd.concat([y_train_new, ytemp])
print(len(X_train_new), len(y_train_new))
from PIL import ImageEnhance
class PreProcessing(object):
    def __init__(self, image = None):
        self.image = image
    def subtract_mean(self):
        self.image = (self.image/255.0 - 0.25)*255
        return self.image
    def downsample_data(self):
        self.image = cv2.resize(self.image, (40,40), interpolation = cv2.INTER_AREA)
        return self.image
    def upsample_data(self):
        self.image = cv2.resize(self.image, (224, 224), interpolation =
cv2.INTER_CUBIC)
        return self.image

```

```

import matplotlib.pyplot as plt
from skimage.io import imread
from PIL import Image
import cv2

dirName = '/content/drive/My Drive/Dataset/train/'
plt.figure(figsize = (10,10))
inp = imread(dirName + 'image_' + str(30517) + '.jpg')
print ("Original shape of input image: ", inp.shape)
plt.subplot(221)
plt.imshow(inp, cmap='gray')
plt.grid(False)
inp = PreProcessing(inp).upsample_data()
Pp = PreProcessing(inp)
inp2 = Pp.subtract_mean()
plt.subplot(222)
plt.imshow(inp2, cmap='gray')
plt.grid(False)
inp3 = ImageEnhance.Contrast(Image.fromarray(inp))
contrast = 1.5
inp3 = inp3.enhance(contrast)
plt.subplot(223)
plt.imshow(inp3, cmap='gray')
plt.grid(False)
inp4 = Pp.downsample_data()
plt.subplot(224)
plt.imshow(inp4,cmap='gray')
plt.grid(False)
y_train_new.values.astype(int)
train_filenames =\
X_train_new.index.to_series().apply(lambda x:\ '/content/drive/My
Drive/Dataset/train/image_'+str(x)+'.jpg')
train_filenames.values.astype(str)
train_filenames.values.astype(str)
dataset_file = 'traindatalabels.txt'

```

```

train_filenames_str = train_filenames.apply(lambda x: [str(filename) for filename in
x])
filenames = train_filenames_str.values
train_filenames = X_train_new.index.to_series().apply(lambda x: filenames)
labels = y_train_new.values.astype(int)
train_filenames = X_train_new.index.to_series().apply(
    lambda x: f'/content/drive/My Drive/Dataset/train/image_{x}.jpg'
)
filenames = train_filenames.values.astype(str)
labels = y_train_new.values.astype(int)
# Prepare data for saving
traindata = pd.DataFrame({
    'filename': filenames,
    'label': labels
})
traindata.to_csv(dataset_file, sep=' ', header=False, index=False, escapechar='\\',
quoting=3)
image_shape = (50, 50) # Desired image shape
output_path = 'dataset.h5'
dataset_file = 'traindatalabels.txt'
base_dir = '/content/drive/MyDrive/Dataset/train/'
data = pd.read_csv(dataset_file, delimiter=' ', header=None, names=['filename',
'label'])
print(data.head())
data['filename'] = data['filename'].apply(lambda x: os.path.join(base_dir,
os.path.basename(x)))
filenames = data['filename'].values
labels = data['label'].values
labels = labels.astype(np.int32)
num_classes = 2
labels_one_hot = np.zeros((len(labels), num_classes), dtype=np.float32)
labels_one_hot[np.arange(len(labels)), labels] = 1
with h5py.File(output_path, 'w') as f:
    X_dset = f.create_dataset('X', (len(filenames), *image_shape), dtype='<f4')

```



```

y_dset = f.create_dataset('y', (len(filenamees), num_classes), dtype='<f4')
for i, (filename, label) in enumerate(zip(filenamees, labels)):
    try:
        if not os.path.isfile(filename):
            raise FileNotFoundError(f"File not found: {filename}")
        image = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
        if image is None:
            raise ValueError(f"Image at {filename} could not be read. It might be
corrupted or unsupported format.")
        image = cv2.resize(image, image_shape, interpolation=cv2.INTER_AREA)
        image = image.astype(np.float32) / 255.0
        X_dset[i, :, :] = image
        y_dset[i, :] = labels_one_hot[i]
    except Exception as e:
        print(f"Error processing image {filename}: {e}")
print("HDF5 dataset created successfully!")
with h5py.File(output_path, 'r') as f:
    print(f['X'].shape) # Should output (number_of_images, 50, 50)
    print(f['y'].shape) # Should output (number_of_images, 2)
with h5py.File('/content/drive/MyDrive/Dataset/traindataset.h5', 'r') as hdf5_file:
    images = hdf5_file['X'][:]
    labels = hdf5_file['Y'][:]

```

```
X_train_processing = np.array(images)
```

```
X_train_processing
```

Building Model

2D CNN1:

```
import numpy as np
```

```
import h5py
```

```
import keras
```

```
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
```

```
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.regularizers import l2
```

```
import tensorflow as tf
```

```
from sklearn.utils import class_weight
```

```

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout, BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import precision_recall_curve, classification_report,
confusion_matrix
import matplotlib.pyplot as plt
with h5py.File('/content/drive/MyDrive/Dataset/traindataset.h5', 'r') as hdf5_file:
    X_train_images_np = np.array(hdf5_file['X'][:])
    y_train_labels_np = np.array(hdf5_file['Y'][:])
with h5py.File('/content/drive/MyDrive/Dataset/val.h5', 'r') as hdf5_file:
    X_val_images_np = np.array(hdf5_file['X'][:])
    y_val_labels_np = np.array(hdf5_file['Y'][:])
with h5py.File('/content/drive/MyDrive/Dataset/testdataset.h5', 'r') as hdf5_file:
    X_test_images_np = np.array(hdf5_file['X'][:])
    y_test_labels_np = np.array(hdf5_file['Y'][:])
if X_train_images_np.ndim == 3:
    X_train_images_np = np.expand_dims(X_train_images_np, axis=-1)
if X_val_images_np.ndim == 3:
    X_val_images_np = np.expand_dims(X_val_images_np, axis=-1)
if X_test_images_np.ndim == 3:
    X_test_images_np = np.expand_dims(X_test_images_np, axis=-1)
def create_model(input_shape=(50, 50, 1)):
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same', input_shape=input_shape))
    model.add(BatchNormalization())
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.1))
    model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same'))
    model.add(BatchNormalization())

```

```

    model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.2))
    model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same'))
    model.add(BatchNormalization())
    model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.3))
    model.add(Flatten())
    model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(2, activation='softmax'))
    return model

def focal_loss(gamma=2., alpha=0.25):
    def focal_loss_fixed(y_true, y_pred):
        epsilon = tf.keras.backend.epsilon()
        y_pred = tf.clip_by_value(y_pred, epsilon, 1. - epsilon)
        y_true = tf.cast(y_true, tf.float32)
        alpha_t = y_true * alpha + (tf.ones_like(y_true) - y_true) * (1 - alpha)
        p_t = y_true * y_pred + (tf.ones_like(y_true) - y_true) * (tf.ones_like(y_pred) -
y_pred)
        fl = - alpha_t * tf.keras.backend.pow((tf.ones_like(y_true) - p_t), gamma) *
tf.keras.backend.log(p_t)
        return tf.reduce_mean(fl)
    return focal_loss_fixed

model = create_model()
model.compile(optimizer='adam', loss=focal_loss(), metrics=['accuracy'])
early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=1,

```

```

restore_best_weights=True)
model_checkpoint = ModelCheckpoint('best_model.keras', monitor='val_accuracy',
save_best_only=True, verbose=1)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5,
verbose=1, min_lr=1e-6)
tf.config.run_functions_eagerly(True)
steps_per_epoch = X_train_images_np.shape[0] // 32
validation_steps = X_val_images_np.shape[0] // 32
def single_input_generator(generator):
    for X_batch, y_batch in generator:
        yield X_batch, y_batch
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    fill_mode='nearest'
)
val_test_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow(X_train_images_np, y_train_labels_np,
batch_size=32, shuffle=True)
val_generator = val_test_datagen.flow(X_val_images_np, y_val_labels_np,
batch_size=32, shuffle=False)
train_multi_generator = tf.data.Dataset.from_generator(
    lambda: single_input_generator(train_generator),
    output_signature=(
        tf.TensorSpec(shape=(None, 50, 50, 1), dtype=tf.float32),
        tf.TensorSpec(shape=(None, 2), dtype=tf.float32)
    )
)
val_multi_generator = tf.data.Dataset.from_generator(

```

```

lambda: single_input_generator(val_generator),
output_signature=(
    tf.TensorSpec(shape=(None, 50, 50, 1), dtype=tf.float32),
    tf.TensorSpec(shape=(None, 2), dtype=tf.float32)
)
)
class_weights = class_weight.compute_class_weight(
    'balanced',
    classes=np.unique(y_train_labels_np[:, 1]),
    y=y_train_labels_np[:, 1]
)
class_weight_dict = {i: weight for i, weight in enumerate(class_weights)}
history = model.fit(
    train_multi_generator,
    steps_per_epoch=steps_per_epoch,
    epochs=500,
    validation_data=val_multi_generator,
    validation_steps=validation_steps,
    class_weight=class_weight_dict,
    callbacks=[early_stopping, model_checkpoint, reduce_lr],
    verbose=1
)
test_generator = val_test_datagen.flow(X_test_images_np, y_test_labels_np,
batch_size=32, shuffle=False)
test_loss, test_accuracy = model.evaluate(test_generator, verbose=1)
print(f"Test Loss: {test_loss:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")

```

2D CNN2:

```

def create_model_2(input_shape=(50, 50, 1)):
    model = Sequential()
    # First Block
    model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same', input_shape=input_shape))

```

```

model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.3))
model.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.4))
model.add(Conv2D(512, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(512, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(256, activation='relu', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

```

```

        model.add(Dense(2, activation='softmax'))
    return model

def focal_loss(gamma=2., alpha=0.25):
    def focal_loss_fixed(y_true, y_pred):
        epsilon = tf.keras.backend.epsilon()
        y_pred = tf.clip_by_value(y_pred, epsilon, 1. - epsilon)
        y_true = tf.cast(y_true, tf.float32)
        alpha_t = y_true * alpha + (tf.ones_like(y_true) - y_true) * (1 - alpha)
        p_t = y_true * y_pred + (tf.ones_like(y_true) - y_true) * (tf.ones_like(y_pred) -
y_pred)
        fl = - alpha_t * tf.keras.backend.pow((tf.ones_like(y_true) - p_t), gamma) *
tf.keras.backend.log(p_t)
        return tf.reduce_mean(fl)
    return focal_loss_fixed

model_2 = create_model_2()
model_2.compile(optimizer='adam', loss=focal_loss(), metrics=['accuracy'])
early_stopping_2 = EarlyStopping(monitor='val_loss', patience=10, verbose=1,
restore_best_weights=True)
model_checkpoint_2 = ModelCheckpoint('best_model_2.keras',
monitor='val_accuracy', save_best_only=True, verbose=1)
reduce_lr_2 = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5,
verbose=1, min_lr=1e-6)
model_2.summary()
history = model_2.fit(
    train_multi_generator,
    steps_per_epoch=len(train_generator),
    epochs=500,
    validation_data=val_multi_generator,
    validation_steps=len(val_generator),
    class_weight=class_weight_dict,
    callbacks=[early_stopping_2, model_checkpoint_2, reduce_lr_2],
    verbose=1
)

```

```
test_loss, test_acc = model_2.evaluate(test_generator, steps=len(test_generator))
print(f"Test loss: {test_loss:.4f}, Test accuracy: {test_acc:.4f}")
```

2D CNN3:

```
def CNN3(input_shape=(50, 50, 1)):
    model = Sequential()
    model.add(Conv2D(32, (3, 3), padding='same', activation='relu',
kernel_initializer='he_uniform', input_shape=input_shape))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu',
kernel_initializer='he_uniform'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(128, (3, 3), padding='same', activation='relu',
kernel_initializer='he_uniform'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu', kernel_regularizer=l2(0.001)))
    model.add(Dropout(0.5))
    model.add(Dense(64, activation='relu', kernel_regularizer=l2(0.001)))
    model.add(Dropout(0.5))
    model.add(Dense(1, activation='sigmoid'))
    return model

third_cnn = CNN3()
optimizer = Adam(learning_rate=1e-5)
def focal_loss(gamma=2., alpha=0.25):
    def focal_loss_fixed(y_true, y_pred):
        epsilon = tf.keras.backend.epsilon()
        y_pred = tf.clip_by_value(y_pred, epsilon, 1. - epsilon)
        y_true = tf.cast(y_true, tf.float32)
        alpha_t = y_true * alpha + (tf.ones_like(y_true) - y_true) * (1 - alpha)
        p_t = y_true * y_pred + (tf.ones_like(y_true) - y_true) * (tf.ones_like(y_pred) -
```



```

y_pred)
    fl = - alpha_t * tf.keras.backend.pow((tf.ones_like(y_true) - p_t), gamma) *
tf.keras.backend.log(p_t)
    return tf.reduce_mean(fl)
return focal_loss_fixed
third_cnn.compile(optimizer=optimizer, loss=focal_loss(), metrics=['accuracy'])
history = third_cnn.fit(
    train_multi_generator,
    steps_per_epoch=steps_per_epoch,
    epochs=500,
    validation_data=val_multi_generator,
    validation_steps=validation_steps,
    class_weight=class_weight_dict,
    callbacks=[early_stopping_3, model_checkpoint_3, reduce_lr_3],
    verbose=1
)
loss, accuracy = third_cnn.evaluate(test_generator, verbose=1)
print(f"Test Loss: {loss:.4f}")
print(f"Test Accuracy: {accuracy:.4f}")

```

Ensemble 2D CNN:

```

train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=20,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   fill_mode='nearest')

val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

batch_size = 32

img_height, img_width = X_train_images_np.shape[1], X_train_images_np.shape[2]
train_generator = train_datagen.flow(X_train_images_np, y_train_labels_np,

```

```

batch_size=batch_size)
val_generator = val_datagen.flow(X_val_images_np, y_val_labels_np,
batch_size=batch_size)
test_generator = test_datagen.flow(X_test_images_np, y_test_labels_np,
batch_size=batch_size, shuffle=False)
y_pred_prob_1 = model_1.predict(test_generator)
y_pred_prob_2 = model_2.predict(test_generator)
y_pred_prob_3 = third_cnn.predict(test_generator)
y_pred_prob_ensemble = (y_pred_prob_1 + y_pred_prob_2 + y_pred_prob_3) / 3
y_pred_ensemble = (y_pred_prob_ensemble > 0.5).astype(int)
if len(y_test_labels_np.shape) > 1 and y_test_labels_np.shape[1] > 1:
    y_test_labels_np = np.argmax(y_test_labels_np, axis=1)
y_pred_ensemble = np.argmax(y_pred_ensemble, axis=1)
cm_ensemble = confusion_matrix(y_test_labels_np, y_pred_ensemble)
print("Ensemble Confusion Matrix:\n", cm_ensemble)
cr_ensemble = classification_report(y_test_labels_np, y_pred_ensemble)
print("Ensemble Classification Report:\n", cr_ensemble)
ensemble_accuracy = np.mean(y_pred_ensemble == y_test_labels_np)
print("Ensemble Accuracy:", ensemble_accuracy)

```

Flask Code to Connect Front End

App.py:

```

from flask import Flask, request, render_template, jsonify
import tensorflow as tf
import numpy as np
import traceback
from PIL import Image
import io
from tensorflow.keras.utils import get_custom_objects
import os
import cv2
app = Flask(__name__)
UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

```

```

if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

try:
    ct_classifier_path = 'Model/ct_vs_nonct_classifier.h5'
    ct_classifier = tf.keras.models.load_model(ct_classifier_path)
    print("✅ CT vs Non-CT classifier loaded successfully")
except Exception as e:
    print(f"❌ Error loading CT classifier model: {e}")
    ct_classifier = None

def focal_loss_fixed(y_true, y_pred):
    gamma = 2.0
    alpha = 0.25
    pt_1 = tf.where(tf.equal(y_true, 1), y_pred, tf.ones_like(y_pred))
    pt_0 = tf.where(tf.equal(y_true, 0), y_pred, tf.zeros_like(y_pred))
    return -tf.reduce_sum(alpha * tf.pow(1. - pt_1, gamma) * tf.math.log(pt_1 + 1e-8))
    - tf.reduce_sum((1 - alpha) * tf.pow(pt_0, gamma) * tf.math.log(1. - pt_0 + 1e-8))
get_custom_objects().update({'focal_loss_fixed': focal_loss_fixed})

try:
    model_1_path = 'Model/improved_model_1 (1).h5' # Update with the correct path
    model_2_path = 'Model/improved_model_2.h5' # Update with the correct path
    model_3_path = 'Model/improved_model_3_updated.h5' # Update with the correct
path
    model_1 = tf.keras.models.load_model(model_1_path,
custom_objects={'focal_loss_fixed': focal_loss_fixed})
    model_2 = tf.keras.models.load_model(model_2_path,
custom_objects={'focal_loss_fixed': focal_loss_fixed})
    model_3 = tf.keras.models.load_model(model_3_path,
custom_objects={'focal_loss_fixed': focal_loss_fixed})
    print("✅ All models loaded successfully")
except Exception as e:
    print(f"❌ Error loading models: {e}")
    model_1, model_2, model_3 = None, None, None

@app.route('/')

```

```

def index():
    return render_template('index.html')
@app.route('/prediction')
def prediction():
    return render_template('Prediction.html')
@app.route('/symptoms')
def symptoms():
    return render_template('symptoms.html')
def is_ct_scan(image):
    try:
        if len(image.shape) == 2:
            image = np.stack((image,) * 3, axis=-1)
            img_resized = cv2.resize(image, (50, 50))
            img_resized = img_resized / 255.0 # Normalize
            img_resized = np.expand_dims(img_resized, axis=0)
            y_pred = ct_classifier.predict(img_resized)
            predicted_class = int(y_pred[0][0] > 0.5) # Assuming sigmoid output (0 = CT, 1
= Non-CT)
            print(f"💎 CT Classifier Prediction: {y_pred[0][0]} | Classified as: {'CT' if
predicted_class == 0 else 'Non-CT'}")
            return predicted_class == 0
        except Exception as e:
            print(f"✖ Error in CT scan classification: {e}")
            return False # Default to rejecting invalid images

@app.route('/predict', methods=['POST'])
def predict():
    if 'file' not in request.files:
        return jsonify({'error': 'No file part'}), 400
    file = request.files['file']
    if file.filename == "":
        return jsonify({'error': 'No file selected'}), 400
    try:
        img = Image.open(io.BytesIO(file.read()))

```

```

image_cv = np.array(img.convert('L')) # Convert to grayscale (H, W)
if not is_ct_scan(image_cv):
    return jsonify({'error': 'Invalid image type. Please upload a Lung CT Scan.'}),
400

img_resized = img.resize((50, 50)).convert('L')
image_array = np.array(img_resized) / 255.0 # Normalize
image_array = np.expand_dims(image_array, axis=(0, -1)) # Shape (1, 50, 50, 1)

y_pred_prob_1 = model_1.predict(image_array)
y_pred_prob_2 = model_2.predict(image_array)
y_pred_prob_3 = model_3.predict(image_array)
y_pred_prob_ensemble = (y_pred_prob_1 + y_pred_prob_2 + y_pred_prob_3) /
3

predicted_class = np.argmax(y_pred_prob_ensemble[0])
# predicted_class = 1 if y_pred_prob_ensemble[0][1] > threshold else 0
confidence = y_pred_prob_ensemble[0][predicted_class] * 100
print(predicted_class, y_pred_prob_ensemble)
class_labels = ['No Cancer', 'Cancer']
predicted_label = class_labels[predicted_class]
response = {
    "predicted": int(predicted_class), # Convert to native Python int
    "label": predicted_label,
    "confidence": round(confidence, 2),
    "probabilities": [float(prob) for prob in y_pred_prob_ensemble[0]] # Ensure
probabilities are float
}

return jsonify(response)
except Exception as e:
    traceback.print_exc()
    return jsonify({'error': 'Error processing the file', 'details': str(e)}), 500
if __name__ == '__main__':
    app.run(debug=False)

```

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Home - Lung Cancer Detection</title>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">
  <!-- Corrected the path for styles.css -->
  <link rel="stylesheet" href="{ { url_for('static', filename='css/styles.css') } }">
  <link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap"
rel="stylesheet">
</head>
<body>
  <header>
    <nav class="navbar">
      <h1 class="logo">Lung Cancer Detection</h1>
      <ul class="nav-links">
        <li><a href="" class="active">Home</a></li>
        <li><a href="prediction">Detection</a></li>
      </ul>
    </nav>
  </header>
  <main>
    <section class="hero">
      <h1>Enhanced Lung Cancer Detection Using Deep Learning Ensemble
Approach.</h1>
      <p>Detect lung cancer using our AI-powered system. Get fast and accurate
predictions.</p>
      <section class="overview">
        <section class="overview">
          <h2>What is Lung Cancer?</h2>
          <p>Lung cancer is one of the leading causes of cancer deaths worldwide. Early detection is
crucial for effective treatment.</p>
          <div class="stats">
            <p><strong>1 in 15</strong> people develop lung cancer</p>
```

```

    <p><strong>50%</strong> increase in survival rate with early detection</p>
</div>

</section>
    <a href="symptoms" class="btn">Get Started</a>
</section>
</main>
<footer>
    <div class="footer-container">
        <div class="team-info">
            <h3>Team Members</h3>
            <ul>
                <li>
                    <strong>Manvitha Ainavolu</strong> -
                    <a
href="mailto:manvithaainavolu@gmail.com">manvithaainavolu@gmail.com</a>
                </li>
                <li>
                    <strong>Naga Revathi Dokku</strong> -
                    <a
href="mailto:nagarevathidokku@example.com">nagarevathidokku@example.com</a>
                </li>
                <li>
                    <strong>Prathima Kasula</strong> -
                    <a
href="mailto:prathimakasula@example.com">prathimakasula@example.com</a>
                </li>
            </ul>
        </div>
        <div class="social-media">
            <h3>Follow Us</h3>
            <ul>
                <li>
                    <a href="https://twitter.com" target="_blank" aria-label="Twitter">
                        <i class="fab fa-twitter"></i>
                    </a>
                </li>
            </ul>
        </div>
    </div>

```

```

        <li>
            <a href="https://linkedin.com" target="_blank" aria-label="LinkedIn">
                <i class="fab fa-linkedin"></i>
            </a>
        </li>
        <li>
            <a href="https://github.com" target="_blank" aria-label="GitHub">
                <i class="fab fa-github"></i>
            </a>
        </li>
    </ul>
</div>
<div class="quick-links">
    <h3>Quick Links</h3>
    <ul>
        <li><a href="/">Home</a></li>
        <li><a href="/prediction">Prediction</a></li>
        <li><a href="/symptoms">Symptoms</a></li>
        <li><a href="/about">About Us</a></li>
    </ul>
</div>
<div class="contact-info">
    <h3>Contact Us</h3>
    <p>Email: <a
href="mailto:info@lungcancerdetection.com">info@lungcancerdetection.com</a></p>
    <p>Phone: +91 9876543210</p>
</div>
</div>
<div class="copyright">
    <p>&copy; 2025 Lung Cancer Detection Project. All rights reserved.</p>
</div>
</footer>
</body>
</html>

```


Prediction.html:

```
<html>
<body>
  <header>
    <nav class="navbar">
      <h1 class="logo">Lung Cancer Detection</h1>
      <ul class="nav-links">
        <li><a href="/">Home</a></li>
        <li><a href="prediction" class="active">Detection</a></li>
      </ul>
    </nav>
  </header>
  <main>
    <section class="prediction-container">
      <h1>Detect Lung Cancer</h1>
      <p>Upload a CT scan image (JPG, JPEG, PNG, MHD, or RAW) to predict
lung cancer.</p>
      <div id="drop-area" class="drop-area">
        <p>Drag & Drop your file here or click to select</p>
        <img id="preview-image" src="" alt="Image Preview" class="hidden">
      </div>
      <div id="file-name-display" class="file-name-display"></div>
      <form id="upload-form" enctype="multipart/form-data">
        <input type="file" id="file-upload" name="file" accept=".jpg, .jpeg, .png,
.mhd, .raw" required hidden>
        <button type="submit" class="btn">Go</button>
      </form>
      <div id="result-box" class="result-box">
        <p id="result-message"></p>
      </div>
    </section>
  </main>
  <script>
    document.addEventListener('DOMContentLoaded', () => {
```

```

const form = document.getElementById('upload-form');
const fileInput = document.getElementById('file-upload');
const dropArea = document.getElementById('drop-area');
const resultMessage = document.getElementById('result-message');
const previewImage = document.getElementById('preview-image');
const fileNameDisplay = document.getElementById('file-name-display');
dropArea.addEventListener('dragover', (event) => {
    event.preventDefault();
    dropArea.classList.add('drag-over');
});
dropArea.addEventListener('dragleave', () => {
    dropArea.classList.remove('drag-over');
});
dropArea.addEventListener('drop', (event) => {
    event.preventDefault();
    dropArea.classList.remove('drag-over')
    const files = event.dataTransfer.files;
    if (files.length) {
        fileInput.files = files;
        displayFileName(files[0].name);
        showImagePreview(files[0]);
    }
});

dropArea.addEventListener('click', () => fileInput.click());
fileInput.addEventListener('change', (event) => {
    const files = event.target.files;
    if (files.length) {
        displayFileName(files[0].name);
        showImagePreview(files[0]);
    }
});
form.addEventListener('submit', async (event) => {
event.preventDefault(); // Prevent form refresh

```

```

if (!fileInput.files.length) {
  resultMessage.textContent = 'Please select a file before submitting.';
  resultMessage.style.color = 'red';
  return;
}

const formData = new FormData();
formData.append('file', fileInput.files[0]);
resultMessage.textContent = 'Processing... Please wait.';
resultMessage.style.color = 'black';

try {
  const response = await fetch('/predict', {
    method: 'POST',
    body: formData,
  });

  const data = await response.json();

  if (!response.ok || data.error) {
    if (data.error.includes("Warning: No reference CT scan found")) {
      resultMessage.textContent = `⚠️ ${data.error}`;
      resultMessage.style.color = 'orange';
    } else {
      resultMessage.textContent = `Error: ${data.error || 'Prediction failed.'}`;
      resultMessage.style.color = 'red';
    }
  } else {
    resultMessage.textContent = `Prediction: ${data.label}.`;
    resultMessage.style.color = 'green';
  }
} catch (error) {
  resultMessage.textContent = `Error: ${error.message}`;
  resultMessage.style.color = 'red';
}

});

function showImagePreview(file) {
  if (!file.type.startsWith('image/')) {

```

```
        resultMessage.textContent = 'Please upload a valid image file.';
        resultMessage.style.color = 'red';
        return;
    }const reader = new FileReader();
    reader.onload = (event) => {
        previewImage.src = event.target.result;
        previewImage.classList.remove('hidden');
    };
    reader.readAsDataURL(file);
}
</script>
</body>
</html>
```

9. RESULT ANALYSIS

2D CNN1:

The model shows strong performance in predicting class 0 (negative class) with high TN and low FP. However, its performance for class 1 (positive class) is moderate, indicated by relatively lower TP and some FN by achieving an accuracy of 95%.

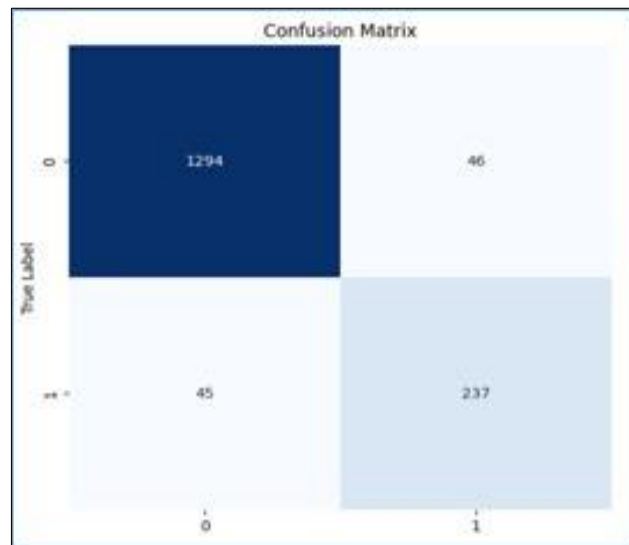


Fig 9.1: Confusion Matrix of 2D CNN1

This confusion matrix shown in fig 9.1 represents the classification performance of the lung cancer detection model. It shows true positives (237), true negatives (1294), false positives (46), and false negatives (45), indicating the model's accuracy in distinguishing between cancerous and non-cancerous cases.

Classification Report for Model 1:				
	precision	recall	f1-score	support
0	0.97	0.97	0.97	1340
1	0.87	0.86	0.87	282
accuracy			0.95	1622
macro avg	0.92	0.92	0.92	1622
weighted avg	0.95	0.95	0.95	1622

Fig 9.2: Classification Report of 2D CNN1

This classification report in the fig 9.2 summarizes the performance of the 2D CNN model, showing precision, recall, and F1-score for each class. The model gave an overall accuracy of 95%, indicating strong performance in distinguishing between cancerous and non-cancerous cases.

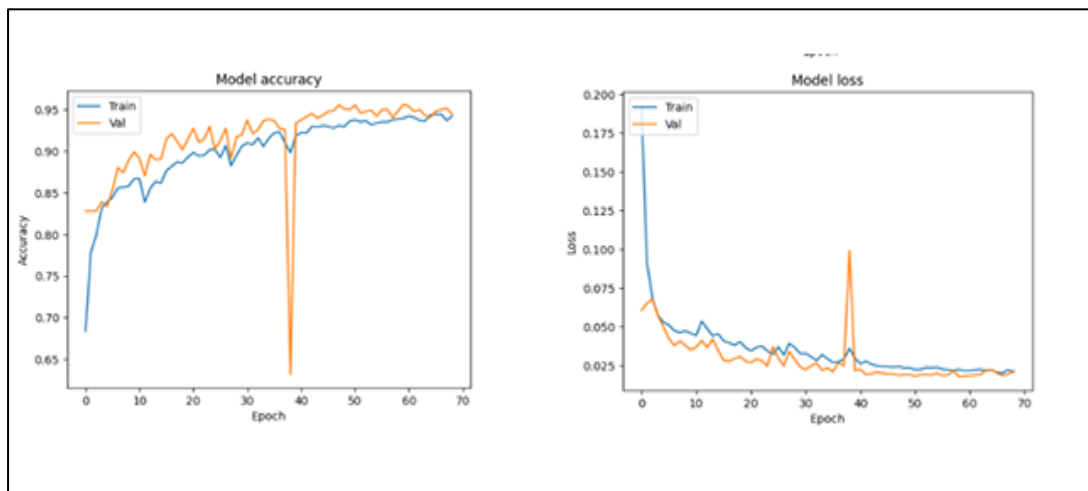


Fig 9.3: Accuracy and Loss Curves of 2D CNN1

The accuracy graph shown in fig 9.3 a steady increase, indicating improved model performance over epochs. The loss curve decreases, demonstrating effective learning. A fluctuation around epoch 40 suggests a temporary instability.

2D CNN2:

This model improves upon the first in correctly predicting class 0, as evidenced by fewer FP. The performance for class 1 remains consistent with the first model, with the same TP and FN values by achieving an accuracy of 94%.

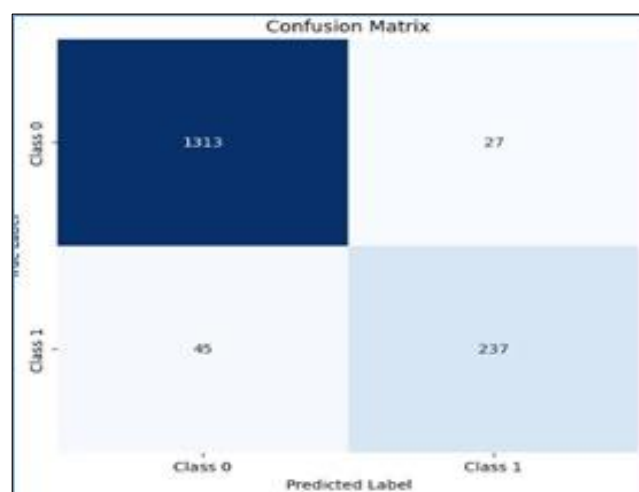


Fig 9.4: Confusion Matrix of 2D CNN2

The confusion matrix shown in fig 9.4 the model's classification performance. It correctly predicts 1313 samples as Class 0 and 237 samples as Class 1, with minor misclassifications.

Classification Report for Model 2:				
	precision	recall	f1-score	support
0	0.95	0.98	0.97	1340
1	0.91	0.78	0.84	282
accuracy			0.95	1622
macro avg	0.93	0.88	0.90	1622
weighted avg	0.95	0.95	0.95	1622

Fig 9.5: Classification report of 2D CNN2

The classification report presented in fig 9.5 gives precision, recall, and F1-score for each class. Model 2 achieves an accuracy of 95%, with Class 0 showing higher recall than Class 1.

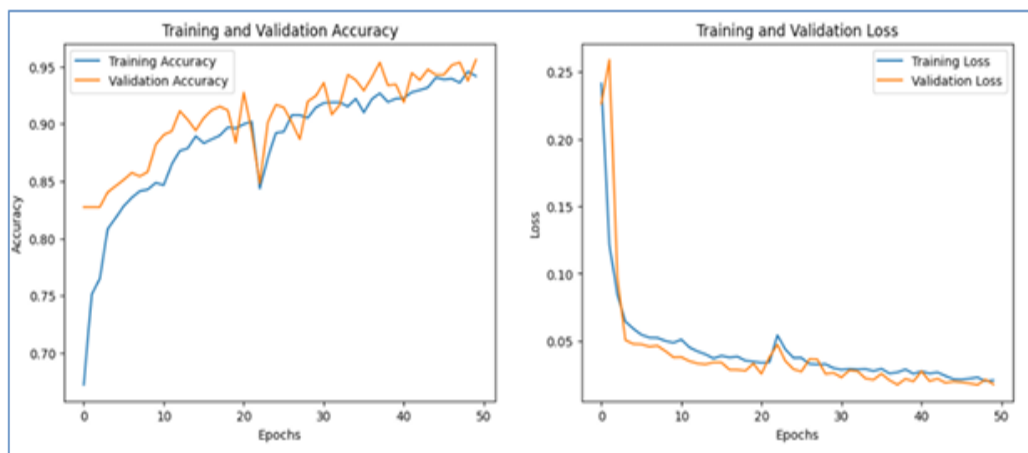


Fig 9.6: Accuracy and Loss Curves for 2D CNN2

The accuracy graph shown in fig 9.6 gives an increasing trend for both training and validation, with minor fluctuations. The loss curve steadily decreases, indicating effective learning.

2D CNN3:

While this model excels at predicting class 0, achieving the highest TN and lowest FP, its performance for class 1 is significantly weaker.

The large number of FN and very few TP indicate that the model struggles to identify positive class instances by achieving an accuracy of 92%..



Fig 9.7: Confusion Matrix for 2D CNN3

The confusion matrix shown in fig 9.7 reports that the model correctly classified 1324 samples as Class 0 and 21 samples as Class 1. However, it misclassified 16 Class 0 samples as Class 1 and 261 Class 1 samples as Class 0, indicating some difficulty in distinguishing Class 1 instances.

Classification Report for Model 3:				
	precision	recall	f1-score	support
0	0.83	1.00	0.90	1340.0
1	0.85	0.82	0.74	282
accuracy			0.93	1622
macro avg	0.78	0.88	0.82	1622
weighted avg	0.78	0.93	0.85	1622

Fig 9.8: Classification Report of 2D CNN3

The classification report for 2D CNN3 shown in fig 9.8 reports an accuracy of 93%, with class 0 achieving high recall (100%) and class 1 having lower recall (82%). The macro average precision, recall, and F1-score indicate some imbalance in class predictions, affecting overall performance.

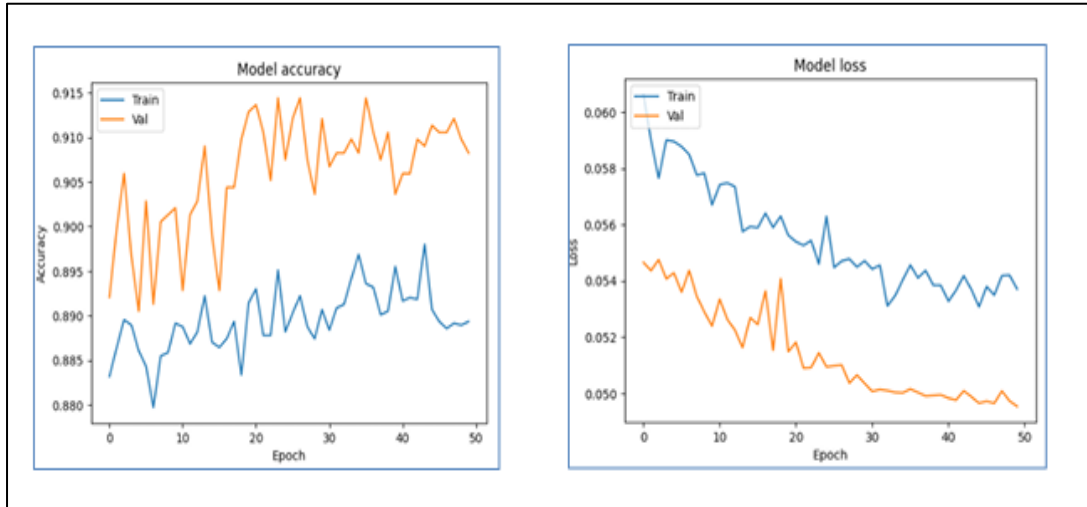


Fig 9.9: Accuracy and Loos Curves for 2D CNN3

The accuracy curves shown in Fig 9.9 reports higher validation accuracy, indicating possible underfitting. The loss curves demonstrate a decreasing trend, though training loss fluctuates, suggesting instability.

Ensemble 2D CNN:

The model achieved an overall accuracy of 96%. For class 0, it demonstrated precision of 0.83 and perfect recall (1.00), resulting in an F1-score of 0.90. For class 1, precision was 0.88 with recall of 0.85, yielding an F1-score of 0.80. The model achieved macro-averaged metrics of 0.78 precision, 0.88 recall, and 0.82 F1-score, while weighted averages were 0.93 for both precision and recall, with 0.85 F1-score.

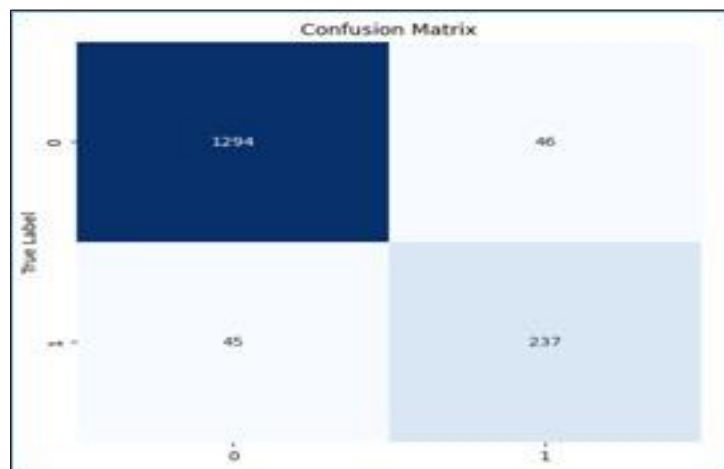


Fig 9.10: Confusion Matrix for Ensemble 2D CNN

The confusion matrix shown in Fig: 9.10 reports 1294 true negatives and 237 true positives, with 46 false positives and 45 false negatives, indicating a well-balanced classification.

Classification Report for Ensemble 2D CNN:				
	precision	recall	f1-score	support
0	0.83	1.00	0.90	1340.0
1	0.88	0.85	0.80	282
accuracy			0.96	1622
macro avg	0.78	0.88	0.82	1622
weighted avg	0.78	0.93	0.85	1622

Fig 9.11: Classification report of Ensemble 2D CNN

The classification report shown in fig: 9.11 gives 96% accuracy, with improved precision (0.88) and f1-score (0.80) for class 1, indicating better performance than the single 2D CNN.

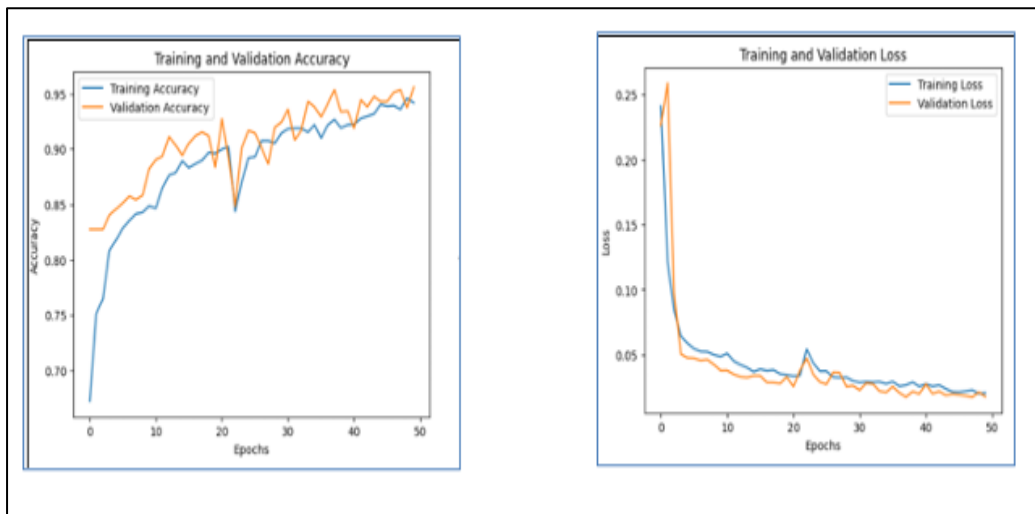


Fig 9.12: Accuracy and Loss Curves for Ensemble 2D CNN

The accuracy curves for the Ensemble 2D CNN shown in Fig: 9.12 steady improvement, surpassing 95%, while the loss curves indicate smooth convergence with minimal overfitting.

The overall analysis of individual CNNs shows varied performance in class prediction, with each model having distinct strengths and limitations. The Ensemble 2D CNN model (96% accuracy) successfully combined their strengths, achieving perfect recall (1.00) for negative class and balanced performance for positive class (0.88 precision, 0.85 recall).

10. TEST CASES

Test case 1: Cancer

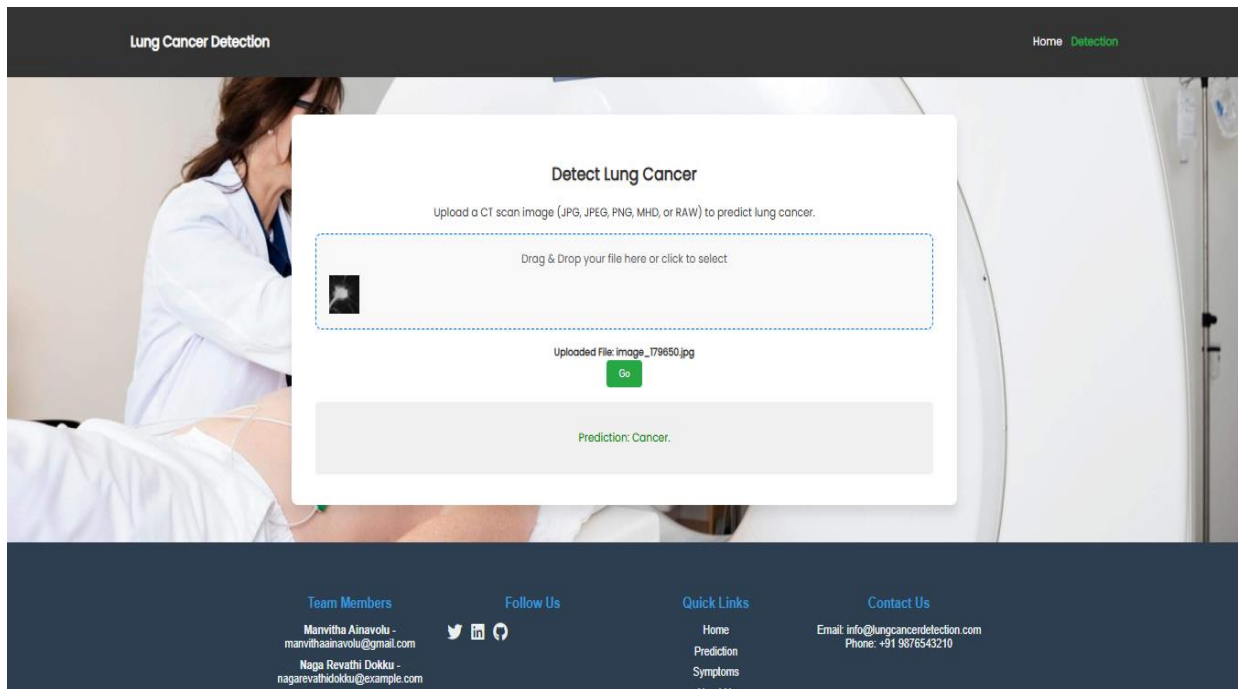


Fig 10.1 Detected Cancer

This image shows the lung cancer detection system where a CT scan image is uploaded. After clicking the "Go" button, the system predicts that the lung is affected by cancer.

Test case 2: Non-Cancer

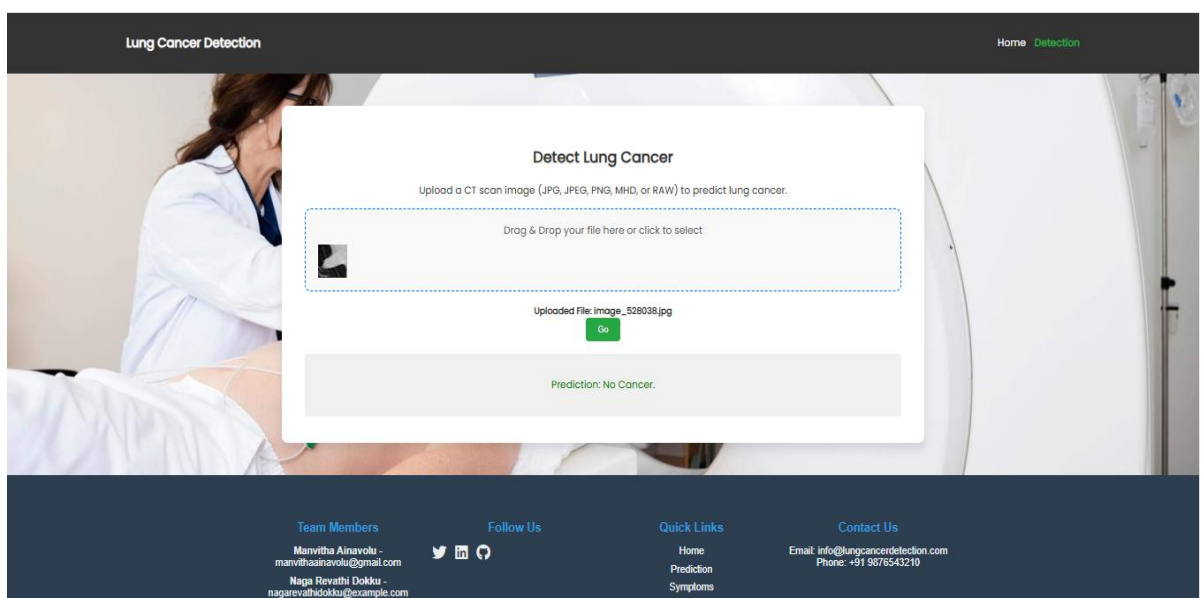


Fig 10.2 Detected Non-Cancer

This image shows the lung cancer detection system where a CT scan image is uploaded. After clicking the "Go" button, the system predicts that the lung is not affected by cancer.

Test Case 3: Validating an input CT scan

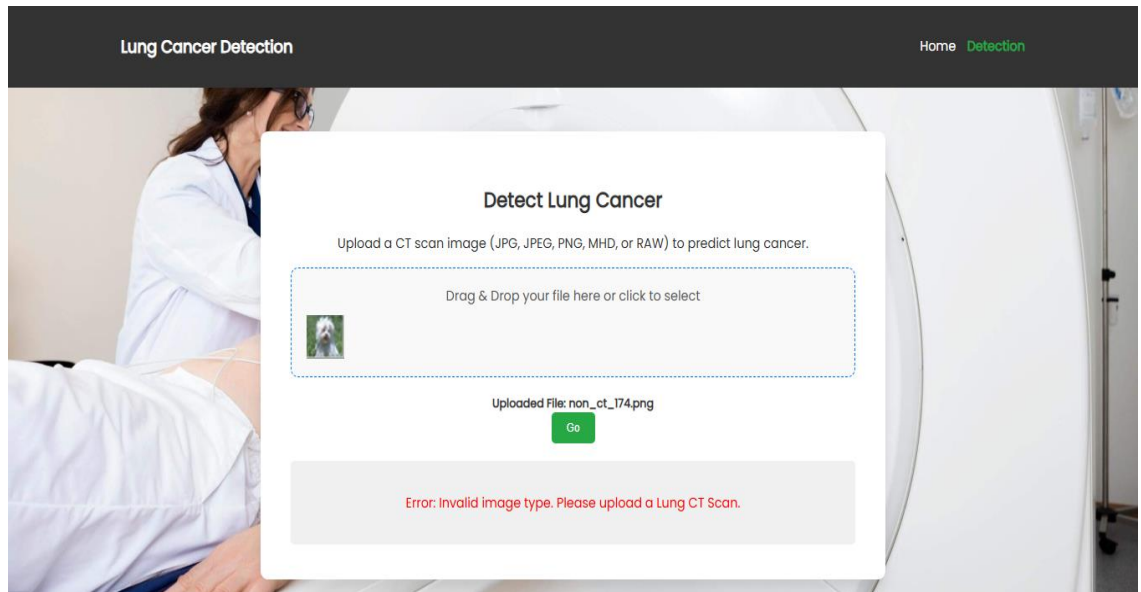


Fig 10.3: Detecting Invalid input Image

In the above figure 10.3 an input image is uploaded that is not a CT Scan of 50x50 size. The model detected that the input image is not a valid CT scan and threw an error when clicked on Go button.

11.USER INTERFACE

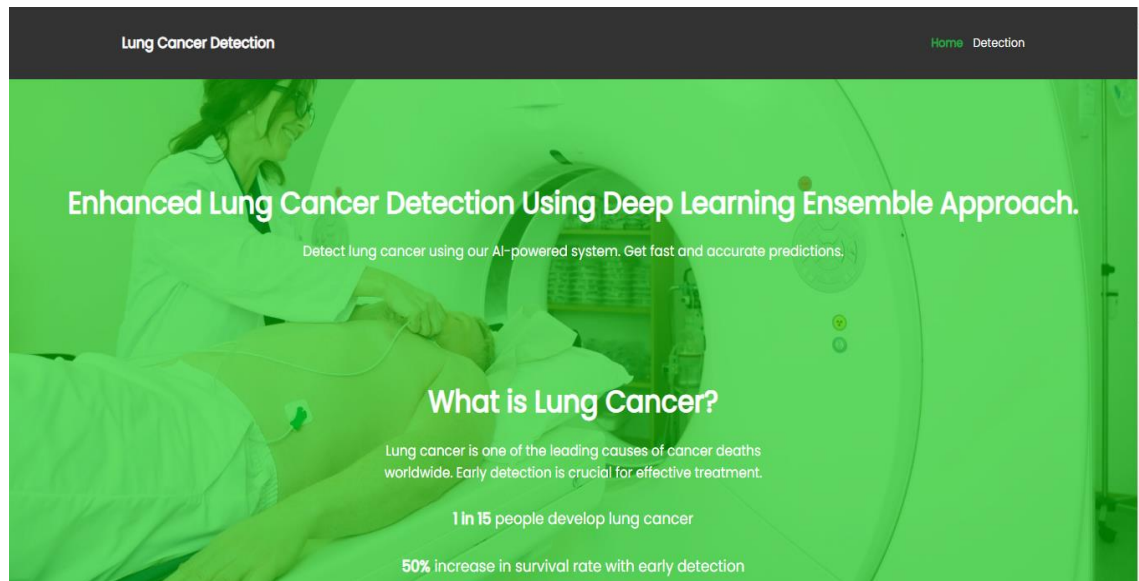


Fig 11.1 Home Screen

This is the home screen shown in fig: 11.1 of the lung cancer detection system. It highlights the use of a deep learning ensemble approach for accurate lung cancer detection and emphasizes the importance of early diagnosis.

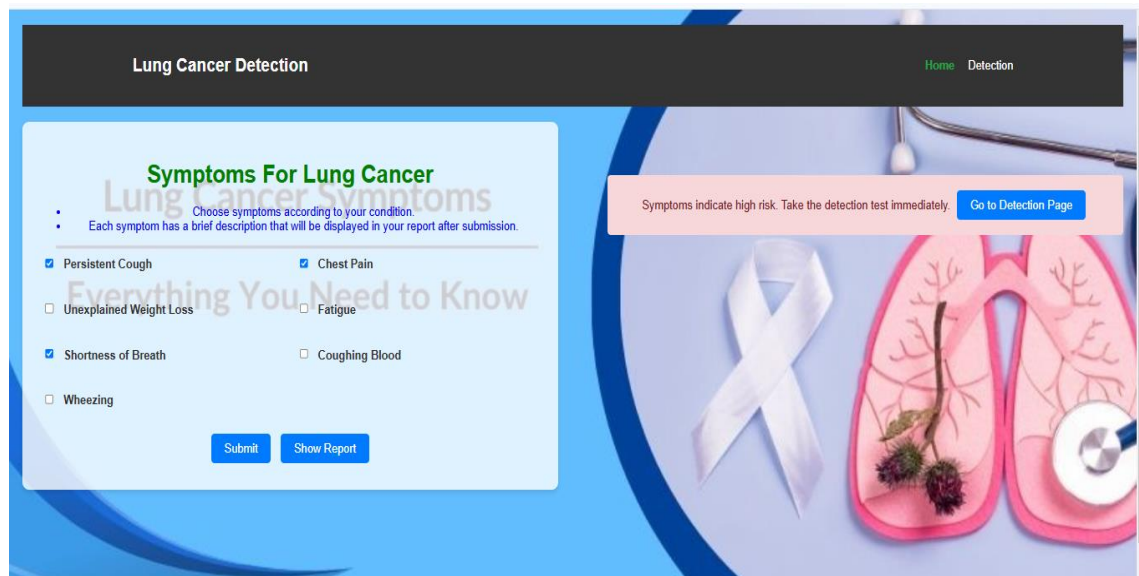


Fig 11.2 Symptoms screen

This screen shown in Fig: 11.2 allows users to select symptoms associated with lung cancer. In this case, the detected symptoms indicate a high risk, prompting the user to take the detection test immediately.

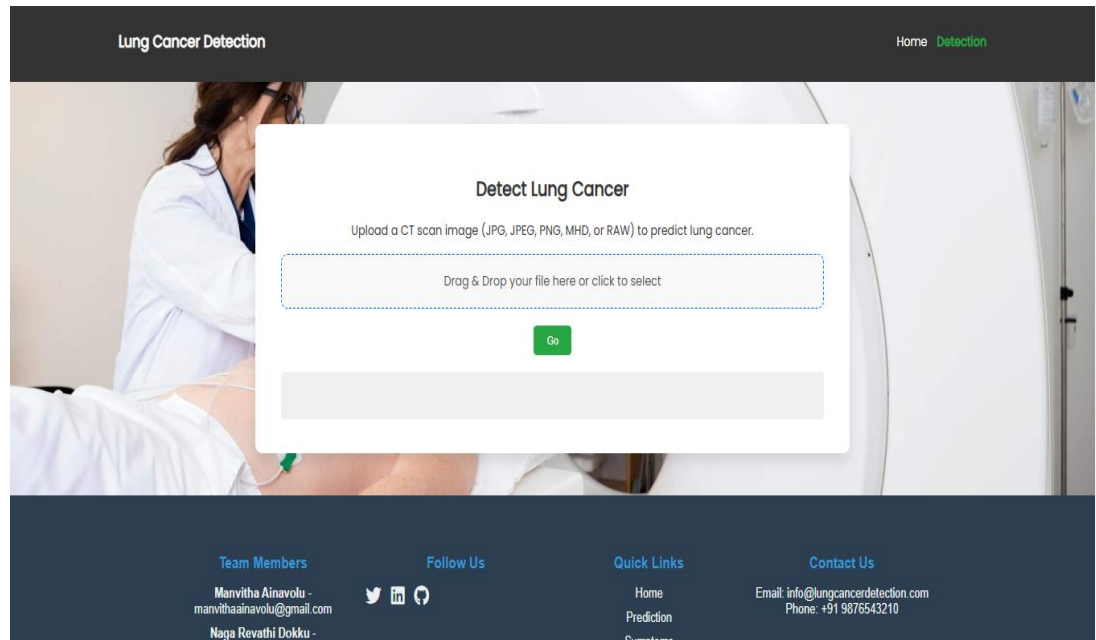


Fig 11.3 Output screen

This screen shown in Fig: 11.3 allows users to upload a CT scan image in various formats (JPG, JPEG, PNG, MHD, or RAW) for lung cancer detection. Once an image is uploaded, the system processes it and provides a prediction regarding the presence of lung cancer.

12.CONCLUSION

In this study, we propose an ensemble learning where three different 2D CNNs are used for lung cancer detection. Each of the three CNNs had different architectures in terms of depth and filter sizes, so diverse features from the dataset could be captured. All the models jointly provided their predictions using a soft voting mechanism to develop an enhanced classification accuracy beyond the capability of a single model. While the first model implemented three convolutional blocks, the second model further deepened the architecture by adding one more block. The third model incorporated different kernel sizes to achieve more complex feature extraction. More significantly, since the classes are imbalanced, two models were trained with Focal Loss to shift the focus onto hard-to classify samples.

This result was based on the ensemble method, with an overall accuracy of 96% on the test set, and the results from individual models were ensembled. Ensemble learning coupled with Focal Loss can achieve great performance improvement in handling class imbalance and obtaining high accuracy compared to traditional loss functions.

13.FUTURE SCOPE

Probably the most relevant limitation of our work resides in the fact that we make use of a 2D CNN model that cannot capture the real three-dimensional characteristics of lung nodules as viewed by CT scans. Moreover, the dataset is comprehensive but somewhat limited in size and diversity, possibly affecting the generalization capability of such a model on other datasets.

Future work involves the exploration of transfer learning techniques for model accuracy and generalization. Transfer learning is a method that allows the pre-trained models to be fine-tuned on our dataset, reaping the knowledge about feature extraction from those models and reducing the training time. This methodology will surely mitigate the limitations brought by smaller datasets using the learned knowledge from larger medical imaging datasets publicly available.

14. REFERENCES

1. Wang, L. (2022). Deep Learning techniques to diagnose lung cancer. *Cancers*, 14(22), 5569. <https://doi.org/10.3390/cancers14225569>
2. Heuvelmans, M. A., van Ooijen, P. M., Ather, S., Silva, C. F., Han, D., Heussel, C. P., ... & Oudkerk, M. (2021). Lung cancer prediction by Deep Learning to identify benign lung nodules. *Lung Cancer*, 154, 14. <https://doi.org/10.1016/j.lungcan.2021.01.027>
3. Li, Z., Zhang, J., Tan, T., Teng, X., Sun, X., Zhao, H., ... & Litjens, G. (2020). Deep Learning methods for lung cancer segmentation in whole-slide histopathology images—the acdc@lunghp challenge 2019. *IEEE Journal of Biomedical and Health Informatics*, 25(2), 429-440. <https://doi.org/10.1109/JBHI.2020.3039741>
4. Huang, P., Lin, C. T., Li, Y., Tammemagi, M. C., Brock, M. V., Atkar-Khattra, S., ... & Lam, S. (2019). Prediction of lung cancer risk at follow-up screening with low-dose CT: a training and validation study of a Deep Learning method. *The Lancet Digital Health*, 1(7), e353-e362. <https://doi.org/10.1109/JBHI.2020.3039741>
5. Subramanian, R. R., Mourya, R. N., Reddy, V. P. T., Reddy, B. N., & Amara, S. (2020). Lung cancer prediction using Deep Learning framework. *International Journal of Control and Automation*, 13(3), 154-160. <https://www.researchgate.net/publication/346700750>
6. Shimazaki, A., Ueda, D., Choppin, A., Yamamoto, A., Honjo, T., Shimahara, Y., & Miki, Y. (2022). Deep Learning-based algorithm for lung cancer detection on chest radiographs using the segmentation method. *Scientific Reports*, 12(1), 727. <https://doi.org/10.1038/s41598-021-04667-w>
7. Lin, C. J., Jeng, S. Y., & Chen, M. K. (2020). Lung Cancer Detection Using 2D CNN with Taguchi parametric optimization for lung cancer recognition from CT images. *Applied Sciences*, 10(7), 2591. <https://doi.org/10.3390/app10025991>
8. Zhou, J., Hu, B., Feng, W., Zhang, Z., Fu, X., Shao, H., ... & Ji, Y. (2023). An ensemble Deep Learning model for risk stratification of invasive lung adenocarcinoma using thin-slice CT. *NPJ Digital Medicine*, 6(1), 119. <https://doi.org/10.1038/s41746-023-00866-z>
9. Tekade, R., & Rajeswari, K. (2018, August). Lung cancer detection and classification using Deep Learning. *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 1-5. <https://doi.org/10.1109/ICCUBEA.2018.8697352>

10. Vijh, S., Gaurav, P., & Pandey, H. M. (2023). Hybrid bio-inspired algorithm and convolutional neural network for automatic lung tumor detection. *Neural Computing and Applications*, 35(33), 23711-23724. <https://doi.org/10.1007/s00521-020-05362-z>
11. Makaju, S., Prasad, P. W. C., Alsadoon, A., Singh, A. K., & Elchouemi, A. (2018). Lung cancer detection using CT scan images. *Procedia Computer Science*, 125, 107-114. <https://doi.org/10.1016/j.procs.2017.12.016>
12. Kalaivani, N., Manimaran, N., Sophia, S., & Devi, D. D. (2020, December). Deep Learning-based lung cancer detection and classification. *IOP Conference Series: Materials Science and Engineering*, 994(1), 012026. <https://doi.org/10.1088/1757-899X/994/1/012026>
13. Shakeel, P. M., Burhanuddin, M. A., & Desa, M. I. (2019). Lung cancer detection from CT image using improved profuse clustering and Deep Learning instantaneously trained neural networks. *Measurement*, 145, 702-712. <https://doi.org/10.1016/j.measurement.2019.05.027>
14. Moturi, S., et al. (2024). Prediction of Liver Disease Using Machine Learning Algorithms. In: Nanda, S.J., Yadav, R.P., Gandomi, A.H., Saraswat, M. (eds) *Data Science and Applications. ICDSA 2023. Lecture Notes in Networks and Systems*, vol 820. Springer, Singapore. https://doi.org/10.1007/978-981-99-7817-5_19
15. Seva, A., Tirumala Rao, S. N., & Sireesha, M. (2024). Prediction of Liver Disease with Random Forest Classifier Through SMOTE-ENN Balancing. *2024 IEEE 13th International Conference on Communication Systems and Network Technologies (CSNT)*, Jabalpur, India, 928-933. <https://doi.org/10.1109/CSNT60213.2024.10546170>

2024 First IEEE International Conference for
**WOMEN IN COMPUTING
(INCOWOCO 2024)**

14 - 15, November 2024 | Pune, Maharashtra, India

CERTIFICATE

This certificate is presented to

WCS
4025



Sireesha Moturi

Dept. of CSE,
Narasaraopeta Engineering College,
Narasaraopeta, Andhra Pradesh

for presenting the research paper entitled "Enhanced Lung Cancer Detection Using Deep Learning Ensemble Approach" in the 2024 First IEEE International Conference for Women in Engineering (INCOWOCO 2024) held at Symbiosis Institute of Computer Studies and Research (SICSR), Symbiosis International (Deemed University), Pune, Maharashtra, India during 14 - 15, November 2024. The conference is technically co-sponsored by IEEE Women in Engineering (WIE) of Pune Section and IEEE Pune Section.

Dr. Priya Gokhale
General Chair

Prof. Dr. Rajashree Jain
General Chair

Dr. Surekha Deshmukh
Honorary Chair

Prof. Dr. Jatinderkumar R. Saini
Honorary Chair

Our Partners



SYMBIOSIS INTERNATIONAL (DEEMED UNIVERSITY)

(Established under Section 3 of the UGC Act, 1956) | Re-accredited by NAAC with 'A++' grade | Awarded Category - I by UGC

2024 First IEEE International Conference for
**WOMEN IN COMPUTING
(INCOWOCO 2024)**

14 - 15, November 2024 | Pune, Maharashtra, India

CERTIFICATE

This certificate is presented to

WCS
4025



Manvitha Ainavolu

Dept. of CSE,
Narasaraopeta Engineering College,
Narasaraopeta, Andhra Pradesh

for presenting the research paper entitled "Enhanced Lung Cancer Detection Using Deep Learning Ensemble Approach" in the 2024 First IEEE International Conference for Women in Engineering (INCOWOCO 2024) held at Symbiosis Institute of Computer Studies and Research (SICSR), Symbiosis International (Deemed University), Pune, Maharashtra, India during 14 - 15, November 2024. The conference is technically co-sponsored by IEEE Women in Engineering (WiE) of Pune Section and IEEE Pune Section.

Dr. Priya Gokhale
General Chair

Prof. Dr. Rajashree Jain
General Chair

Dr. Surekha Deshmukh
Honorary Chair

Prof. Dr. Jatinderkumar R. Saini
Honorary Chair

Our Partners



SYMBIOSIS INTERNATIONAL (DEEMED UNIVERSITY)

(Established under Section 3 of the UGC Act, 1956) | Re-accredited by NAAC with 'A++' grade | Awarded Category - I by UGC

2024 First IEEE International Conference for
**WOMEN IN COMPUTING
(INCOWOCO 2024)**
14 - 15, November 2024 | Pune, Maharashtra, India

CERTIFICATE

This certificate is presented to

WCS
4025



Naga Revathi Dokku

Dept. of CSE,
Narasaraopeta Engineering College,
Narasaraopeta, Andhra Pradesh

for presenting the research paper entitled "Enhanced Lung Cancer Detection Using Deep Learning Ensemble Approach" in the 2024 First IEEE International Conference for Women in Engineering (INCOWOCO 2024) held at Symbiosis Institute of Computer Studies and Research (SICSR), Symbiosis International (Deemed University), Pune, Maharashtra, India during 14 - 15, November 2024. The conference is technically co-sponsored by IEEE Women in Engineering (WiE) of Pune Section and IEEE Pune Section.

Dr. Priya Gokhale
General Chair

Prof. Dr. Rajashree Jain
General Chair

Dr. Surekha Deshmukh
Honorary Chair

Prof. Dr. Jatinderkumar R. Saini
Honorary Chair

Our Partners



SYMBIOSIS INTERNATIONAL (DEEMED UNIVERSITY)

(Established under Section 3 of the UGC Act, 1956) | Re-accredited by NAAC with 'A++' grade | Awarded Category - I by UGC

2024 First IEEE International Conference for
**WOMEN IN COMPUTING
(INCOWOCO 2024)**

14 - 15, November 2024 | Pune, Maharashtra, India

CERTIFICATE

This certificate is presented to

WCS
4025



Prathima Kasula

Dept. of CSE,
Narasaraopeta Engineering College,
Narasaraopeta, Andhra Pradesh

for presenting the research paper entitled "Enhanced Lung Cancer Detection Using Deep Learning Ensemble Approach" in the 2024 First IEEE International Conference for Women in Engineering (INCOWOCO 2024) held at Symbiosis Institute of Computer Studies and Research (SICSR), Symbiosis International (Deemed University), Pune, Maharashtra, India during 14 - 15, November 2024. The conference is technically co-sponsored by IEEE Women in Engineering (WiE) of Pune Section and IEEE Pune Section.

Dr. Priya Gokhale
General Chair

Prof. Dr. Rajashree Jain
General Chair

Dr. Surekha Deshmukh
Honorary Chair

Prof. Dr. Jatinderkumar R. Saini
Honorary Chair

Our Partners



SYMBIOSIS INTERNATIONAL (DEEMED UNIVERSITY)

(Established under Section 3 of the UGC Act, 1956) | Re-accredited by NAAC with 'A++' grade | Awarded Category - I by UGC

2024 First IEEE International Conference for
**WOMEN IN COMPUTING
(INCOWOCO 2024)**

14 - 15, November 2024 | Pune, Maharashtra, India

CERTIFICATE

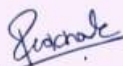
This certificate is presented to



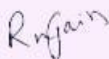
Dodda Venkata Reddy

Dept. of CSE,
Narasaraopeta Engineering College,
Narasaraopeta, Andhra Pradesh

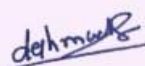
for presenting the research paper entitled "Enhanced Lung Cancer Detection Using Deep Learning Ensemble Approach" in the 2024 First IEEE International Conference for Women in Engineering (INCOWOCO 2024) held at Symbiosis Institute of Computer Studies and Research (SICSR), Symbiosis International (Deemed University), Pune, Maharashtra, India during 14 - 15, November 2024. The conference is technically co-sponsored by IEEE Women in Engineering (WIE) of Pune Section and IEEE Pune Section.



Dr. Priya Gokhale
General Chair



Prof. Dr. Rajashree Jain
General Chair



Dr. Surekha Deshmukh
Honorary Chair



Prof. Dr. Jatinderkumar R. Saini
Honorary Chair

Our Partners



SYMBIOSIS INTERNATIONAL (DEEMED UNIVERSITY)

(Established under Section 3 of the UGC Act, 1956) | Re-accredited by NAAC with 'A++' grade | Awarded Category - I by UGC

2024 First IEEE International Conference for
**WOMEN IN COMPUTING
(INCOWOCO 2024)**

14 - 15, November 2024 | Pune, Maharashtra, India

CERTIFICATE

This certificate is presented to

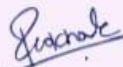
WCS
4025



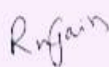
Yaragani Neelima

Dept. of CSE,
Narasaraopeta Engineering College,
Narasaraopeta, Andhra Pradesh

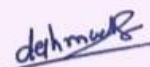
for presenting the research paper entitled "Enhanced Lung Cancer Detection Using Deep Learning Ensemble Approach" in the 2024 First IEEE International Conference for Women in Engineering (INCOWOCO 2024) held at Symbiosis Institute of Computer Studies and Research (SICSR), Symbiosis International (Deemed University), Pune, Maharashtra, India during 14 - 15, November 2024. The conference is technically co-sponsored by IEEE Women in Engineering (WIE) of Pune Section and IEEE Pune Section.



Dr. Priya Gokhale
General Chair



Prof. Dr. Rajashree Jain
General Chair



Dr. Surekha Deshmukh
Honorary Chair



Prof. Dr. Jatinderkumar R. Saini
Honorary Chair

Our Partners



SYMBIOSIS INTERNATIONAL (DEEMED UNIVERSITY)

(Established under Section 3 of the UGC Act, 1956) | Re-accredited by NAAC with 'A++' grade | Awarded Category - I by UGC

Enhanced Lung Cancer Detection Using Deep Learning Ensemble Approach

Sireesha Moturi, Manvitha Ainavolu, Naga Revathi Dokku, Prathima Kasula,
Neelima Yaragani, Venkatareddy Dodda

Department of Computer Science and Engineering,

Narasaraopeta Engineering College, Palnadu Dist., Andhra Pradesh, India

sireeshamoturi@gmail.com, manvithaainavolu@gmail.com, nagarevathidokku@gmail.com,

prathimakasula5@gmail.com, neelimayaragani@gmail.com, oddavenkatareddy@gmail.com

Abstract—In emerging technologies, deep learning has become the solution for most real-world problems. This is used by integrating with computer science and predicting many diseases for people at an early stage. One of the major issues is the detection of Lung Cancer either Cancerous or Non-Cancerous. Many studies have proposed the solutions to detect the Lung Nodule. However, using an ensemble approach by taking CT(Computed Tomography) Scan images gives accurate results when compared to other methodologies. In this study, we proposed the methodology by combining three deep learning models instead of one. Thereby, the performance of three Convolutional Neural Network models is combined to get accurate results. The dataset used for this is LUNA16 from Grand Challenge which is available online. This dataset consists of CT scans and an annotations file which gives the information about each CT Scan. Our Ensemble 2D CNN model gave an effectiveness of 96% which is greater than the baseline methodology.

Index Terms—Deep Learning, Lung Cancer, Computed Tomography Scan, 2D Convolutional Neural Networks, Lung Nodule Analysis 2016, Ensemble Approach, Cancerous Nodules, Non-Cancerous Nodules.

I. INTRODUCTION

Lung Cancer has become the most frequent disease in these days. Many people in the world are suffering from this problem for not identifying this disease early. The development of technology has also increased many solutions in the medical industry. However, no solution detects lung cancer through the individual symptoms. Mostly detecting through medical images is a quite difficult task to identify whether the person is prone to cancer or not. The United States has registered the greatest number of cases worldwide. In the year 2018, the GLOBOCAN estimated that nearly 2 million new cases were registered due to lung cancer.[1]

Deep learning has transformed many sectors of the economy, most importantly medical imaging, providing an automated and much more accurate means for identifying such hard-to-diagnose conditions as lung cancer. Deep learning method called Convolutional Neural Networks (CNNs) holds a highly promising ability in interpreting CT scans and determining if the lung nodule is malignant or not. This will give a promising result to find the state of a person quickly [2]. A CNN can be trained to recognize patterns in large medical image datasets that are usually invisible to

the human eye. Such a CNN trained with thousands of CT scans can be trained to discern subtle differences between benign and malignant nodules, offering an essential early detection approach.

For example, a small nodule might not give any symptoms in the patient, so clinical evaluation would be pointless. Only through CT scans, can these nodules be seen by radiologists. Or interpretation of images by humans is a time-consuming subjective activity and even leads to errors. Small benign nodules, for example, are often mistaken for malignant or overlooked, leading to false positives and unnecessary treatments. Another important task in classifying these images is handling three-dimensional images. The medical images are visualized under 3D view to analyze clearly. So to handle these types of images needs to be efficient in identifying them. DL approaches introduce Neural networks which will learn the patterns of images in all dimensions. Deeper CNNs like GoogleNet, AlexNet, and ResNet [3] became advanced in analyzing the input patterns. From this, we have chosen CNNs as it has a multi-view architecture in processing input images. However, there are some disadvantages to this. Only certain features of lung nodules are identified, which leads to incorrect prognosis and incorrect classification [4].

In this study, we have improved the CNN to classify the lung Nodules. For this purpose, we have chosen LUNA16 from the Grand Challenge. It is an important task when learning the data for the model. Neural networks work like the human brain, so training the data for the model is crucial because it plays a valuable role in the model's accuracy and performance.[5] Here the ensemble approach is developed since it is tough to distinguish between a nodule and a tissue. The main goal in this is to combine the efficiencies of three CNN models and give the result accurately.

II. RELATED WORK

A study conducted in 2018 revealed that a deep learning system could point out overlooked tumors of the lung on chest X-rays, reducing unnecessary CT scans caused by it. Previous studies documented a deep learning-based auto-detection approach to reduce ignored lung tumors on chest radiography.[6] Another study also used 2D CNNs and

Taguchi parametric optimization, but the marginal adjustment of the important parameters increased in this context to improve the percentages of identification of lung cancer from the CT images [7].

The other explained how to take the advantage of the benefits by combining several deep learning models such as CNNs and RNNs.[8] CNNs and deep neural networks were found to have great performances for malignant lung nodule classification with a log loss of 0.387732.

A variant of work introduced a bio-inspired algorithm combined with CNN for lung tumor detection. Using the WOAAPSO algorithm in this hybrid mechanism gave good results with 97.18 % accuracy, sensitivity of 97%, and specificity up to 98.66 standard [10]. The combined results provided evidence that CT over CAD enhances detection, though even the current methods are at fault-perfect to this day still requiring 100% accuracy[11]

Some reviews say that AlexNet, when combined with various classifiers, offers accuracy as high in the detection of lung cancer, while some say that VGG-16, ResNet, and Inception provide the best performance.[12] Deep learning methods, especially CNN, also performed well in classifying lung cancer subtypes, such as adenocarcinoma and squamous cell carcinoma, from pathology images. In some cases, these reached 0.97 AUC, almost the same accuracy as human pathologists could achieve [13]

One of the research study proposes the ensemble model that combines the outputs of different deep-learning architectures: CNNs and RNNs. The strengths of several models are combined using the ensemble approach in order to enhance robustness and generalizability.[14] Some literature also shows that the addition of various types of classifiers along with AlexNet improves the detection accuracy in lung cancer, although various other studies mention that models such as VGG-16, ResNet, and Inception provide the best performance; the accuracy may even reach 100% in some. On the whole, it has been observed that the CNN-based architecture Inception V3 gave very impressive results[15]

III. PROPOSED METHOD

The use of group learning methods has not been relatively easy in previous studies. In this work, we present the use of three different 2D CNN ensembles to identify lung cancer using computed tomography scans. Each of these CNNs was designed to capture a host of features by having different architectural depths and filter sizes. Each model will be trained separately with the aim of increasing the general accuracy of prediction by subsequently combining them using soft voting. A soft voting mechanism would apply to combine the predictions of each model in order to achieve high overall prediction accuracy. The method does this by averaging the probability outputs from all the models for a given instance to decide which class contains the highest average probability. This technique enhances generalization on unseen data by reducing model variance while simultaneously increasing the robustness of predictions.

The followings are the steps involved in the recommended approach:

- Data Description
- Pre-processing.
- Data Splitting.
- Creating the Model Architecture.
- Training the Model Train Dataset
- Testing the Model Using the Test Dataset

The Figure-1 shows the flowchart describes the research paradigm in this study:

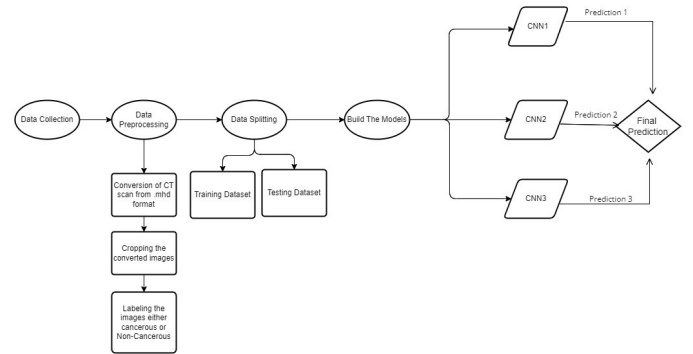


Fig. 1. Flow chart for the Proposed Methodology

A. Experimental Setup

All models were trained using the Adam optimizer, and the batch size was set to 32 for all models. The initial learning rate was set to 1×10^{-3} . All experiments were conducted on an NVIDIA Tesla V100 GPU with 32 GB of RAM. On average, it took approximately 6 hours to train each model. The softmax outputs of the three models were averaged to apply the ensemble approach.

B. Data Collection

Lung Nodule Analysis Dataset 2016 (LUNA16) is an important source for investigators involved in lung nodule detection. This dataset is a subset of a larger database, the LIDC/IDRI, which includes 1,018 CT scans that were reviewed by four expert radiologists.[4] The LIDC/IDRI database classifies the nodules into three classes: nodules 3 mm or larger, nodules smaller than 3 mm, and non-nodules. Last, it comprises annotations indicating the centroid locations and the diameters of detected pulmonary nodules for all of its chest CT images counting to 888. To keep results consistent, users should use these subsets in 10-fold cross-validation. Each of these contains CT images along with several key files such as annotations.csv (reference annotations), sampleSubmission.csv (an example submission format), candidates.csv (basic candidate sites), candidates_V2.csv (extended candidate sites), evaluation scripts, lung segmentation data, additional annotations.csv.

The Dataset can be accessed from LUNA16(Lung Nodule Analysis 2016) Grand Challenge, it has become a widely used source for research, especially on pulmonary nodule detection and related fields.

C. Data Preprocessing

The dataset used in this study is the LUNA16 sourced from Grand Challenge 2016. This dataset contains data about lung cancer with the CT scan images in .mhd format.

- .mhd files contain image resolution, pixel spacing, and data type metadata. This information correctly interprets the associated .raw file, which contains the raw pixel data.
- This is done with libraries such as SimpleITK. The images after converting from .mhd to JPEG format are visualized, and Figure 2 shows the images

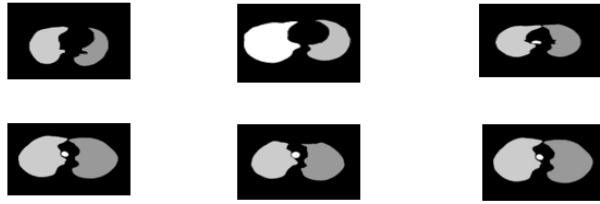


Fig. 2. CT scan images before Pre processing

- Once the image is reconstructed, the image can be saved in .JPEG format using a standard image processing library PIL (Python Imaging Library). After converting the images from MHD to JPEG format and applying normalization.
- The images in JPEG format are cropped in the region
- based on the coordinates given in the annotations.csv file, which gives information about each CT scan. Figure 3 shows the preprocessed images.

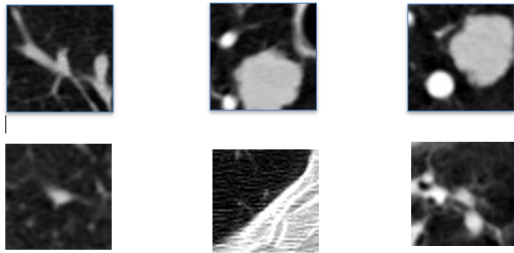


Fig. 3. Cropped CT Scan images

D. Splitting the Dataset

Creating train and test folders out of the dataset is the next step. The candidates.csv file's class column is used for the splitting process. Each CT scan is represented in this column according to its cancerous(1) or non-cancerous(0) status. The dataset is divided into two distinct sub-folders,

one labeled "Cancerous" and the other "Non-Cancerous," within the data folder. The model learns about the data using the train set of data. The real data that we use to evaluate the predictions and accuracy of the model is called test data.

E. Model Architecture and Training

These are the model architecture construction and training phases, which are critical upon completion of the splitting process. The Ensemble 2D CNN algorithm is the one we have selected. We have trained three 2D CNN models in order to apply this Ensemble approach, having a few layers, and join the predictions of models in order to obtain the ensemble outcomes. The description of the three model architectures is as follows:

• Ensemble 2D CNN

The building of model architectures for the three CNN networks are described below.

CNN1: In designing the first model of the convolutional neural network, a three-convolutional-block architecture was followed. In each convolution block, an additional batch normalization layer, max-p pooling layer, and dropout layer were added. Among them, the convolution layers use the ReLU activation function and size filters of 32, 64, and 128, respectively, with a 3x3 kernel size. This will enable the 'he_uniform' initializer to optimize weights for ReLU activation and set weights to small, random values. Following flattening, the model adds a fully linked layer with 128 units using ReLU as an activation function. Next comes the thick layer with softmax activation that gives class probabilities. It uses Adam optimizer for optimization and focused loss while training to handle the issue of class imbalance.

CNN2: The second model deepens the architecture by the introduction of a fourth convolution block, scaling the depth of features to be extracted from 64 to 512 filters. Each convolution block introduces a batch normalization layer, and the next layers are the max-pooling layer and a dropout layer. The dropout rate increases progressively from 0.2 to 0.5. The activation function remains ReLU for hidden layers, with the initializer as 'he_uniform' for optimizing weight distribution. The fully connected ending layer is of dimension 256 units, the activation ReLU. The softmax function will be used for the final output layer to give class probabilities. In this model, Focal loss is similar to the first CNN model.

CNN3: The third model is a bit more complex, which will define a CNN architecture with three convolutional blocks followed by batch normalization, ReLU, and dropout layers, respectively. The first two convolution blocks will contain 32 and 64 filters, each with kernel sizes 3x3 and 5x5, respectively, while the third block has 128 with kernel size 3x3. Kernel size 3x3. After that, the model is terminated by a flattening layer, which is followed by a fully connected layer of units 84, activated by ReLU. Lastly, for output, softmax activation has been used to handle binary classification problems.

Each model produced a probability vector after training over the probability of the input image being in either of the two classes. During inference, these vectors from all three models contain the operations, they are represented as

- **Convolution Operation**

This applies small filters to the input image. It is mathematically represented as:

$$O(i, j) = \sum_{m=U}^{F-1} \sum_{n=U}^{F-1} W(a + g, b + h) \cdot K(g, h) \quad (1)$$

Where:

- F is the size of the filter
- W is the input image
- K is the convolutional kernel
- O denotes the output feature map
- g, h are the indices for the kernel
- a, b are the spatial indices of the feature map

- **ReLU Activation Function**

This introduces non-linearity to the model and mitigates the vanishing gradient problem. It is defined as:

$$\text{ReLU}(y) = \max(0, y) \quad (2)$$

Where:

- y is the argument of the ReLU function

- **Max Pooling**

This is used to downsample the feature maps. It is denoted as:

$$\text{Max Pooling}(p, q) = \max(\text{Window}(p, q)) \quad (3)$$

Where:

- p, q represent the position or index of the window in the feature map

- **Fully Connected Layer (Dense Layer)**

This is the final stage of feature learning. It is defined as:

$$P(y = c|z) = \frac{e^{z_c}}{\sum_{k=1}^C e^{z_k}} \quad (4)$$

Where:

- C is the number of classes
- z_c is the logit for class c

- **Focal Loss**

This function helps handle class imbalance during training. It is given by:

$$FL(a_b) = -\alpha_t(1 - a_b)^\gamma \log(a_b) \quad (5)$$

Where:

- a_b is the predicted probability
- α_t is the weighting factor
- γ is the focusing parameter

- **He Uniform Initialization**

This is also known as He Uniform Initialization, a weighting technique, defined as:

$$\text{Limit} = \sqrt{\frac{6}{n_{in}}} \quad (6)$$

Where:

- n_{in} is the number of input units to the layer

- **Soft Voting**

The predictions from the three models are combined using soft voting, defined as:

$$P_c = \frac{1}{M} \sum_{m=1}^M P_{m,c} \quad (7)$$

Where:

- M is the number of models
- $P_{m,c}$ is the predicted probability for model m and class c

1) 3D CNN

: In the architecture of the third CNN, there will be a series of convolutional blocks, each of which gradually extracts increasingly intricate information from input images. Each of these has three main components: convolutional layers, dropout and batch normalization, and ReLU activation is applied at every stage.

The first convolutional block comes with a kernel size of 3×3 and 32 filters. It captures detailed low-level information such as edges and textures. The second convolutional block has the increased kernel size to 5×5, and the number of filters is increased up to 64. This additional block assists in more complex structure and pattern detection. In the final third convolutional block, it increases the number of filters to 128 with a 3×3 kernel size to focus more on even finer and detailed features.

The output of such convolutional layers is passed through a flattening layer that takes this multi-dimensional output and compresses it into a 1D vector. This vector then inputs a fully connected layer of size 84, where the ReLU activation function assists the model to find non-linear relations in data. Ultimately, whether the picture belongs to the first or the second category-for example, cancerous or not, which is the case if we want to classify tissue as either cancerous or non-cancerous-in general, depends on the output of a softmax activation function that produces probabilities for binary classification.

Each block uses dropout to help with overfitting, making certain neurons randomly switch off during training and utilizing the use of batch normalization that normalizes each layer's output in an effort to stabilize the training and speed up the convergence. Combining all these methods helps to improve the generalizing capacity of the model. It is much more helpful while dealing with intricate datasets like medical imaging where it is pretty important to identify even slighter patterns.

F. Experimental Results and Analysis

The study bases several criteria to assess the efficiency of the proposed ensemble deep learning model. These are F1-score, recall, accuracy, and precision. Each of these

measures was selected to evaluate different facets of functionality of the model ensuring a multi-faceted perception of its strengths and weaknesses.

1) *Results for ensemble 2D CNN model::* The above model architectures were compiled and trained on the dataset. The results and analysis of the models were described below: **CNN1:** The model's overall effectiveness was quite high, at an accuracy of 95%, but with good precision on non-cancerous cases classification at 0.98. The model is trained for 70 epochs. In the case of the cancerous cases, this recall was higher at 0.91, meaning that more true positives were likely to be caught by the model, but at the expense of a higher rate of false positives. The F1-score for the cancerous class showed an instance balance between precision and recall at 0.82. The Figure 4 and figure 5 shows the results and ROC curves of CNN1 are shown below:

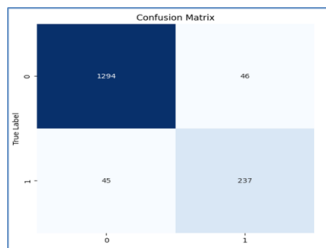


Fig. 4. Confusion matrix for CNN1

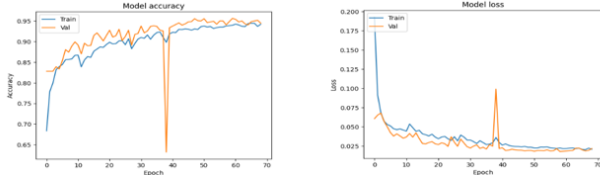


Fig. 5. Accuracy and Loss Curves for CNN1

CNN2 The CNN2 model performed very well in differentiating cancerous from non-cancerous cases, reflected by a testing accuracy of 94.91%. The high precision and recall values for both classes demonstrate that the model is good in predicting cancerous cases. However, it tends to have a bit more false negatives, 45 cases, compared with false positives, 27 cases. The figure 6 and figure 7 shows the results and ROC curves of CNN2 are shown below:

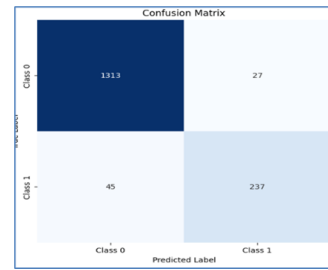


Fig. 6. Confusion matrix for CNN2

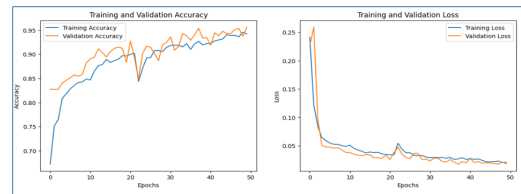


Fig. 7. Accuracy and Loss curves for CNN2

CNN3 Subsequently, the model was used in testing data to estimate the result of the CNN in line with the results obtained from the study. The first model of CNN in iteration shows us an accuracy of 91%. The model was trained with 50 epochs. Figure 8 and Figure 9 shows the the results and ROC curves of CNN3 are shown below:

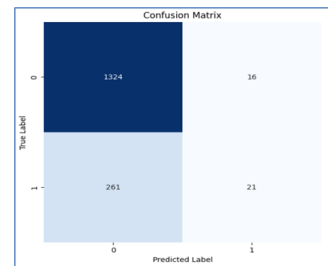


Fig. 8. Confusion Matrix for CNN3

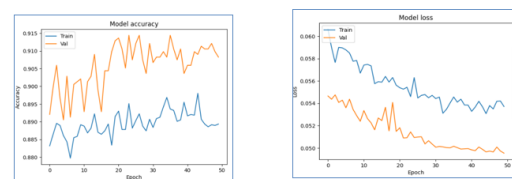


Fig. 9. Accuracy and loss curves for CNN3

Results for ensemble 2D CNN Constructing the ensemble model will combine the probabilities of those three models. Performances will combined because the models were assembled by using different input layers and activation functions. This approach has been improved the performance of the resultant model, as the miscalssification of nodules is reduced.

Three separate models had accuracy values that were around

that point. Once we combined the three models, our accuracy was 96%. This is because an ensemble has the added advantage of making strong, reliable predictions by learning a variety of patterns from the dataset without overfitting.

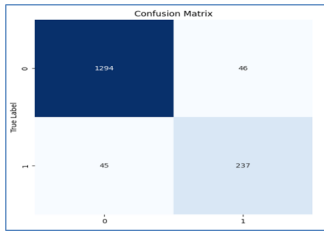


Fig. 10. Confusion Matrix for 2D CNN

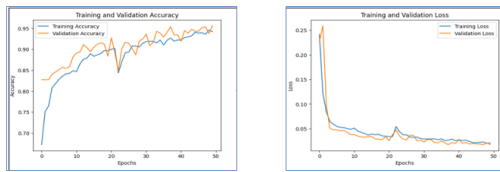


Fig. 11. Accuracy and loss Curves for Ensemble 2D CNN

Classification Report for Model 1:

	precision	recall	f1-score	support
0	0.97	0.97	0.97	1340
1	0.87	0.86	0.87	282
accuracy			0.95	1622
macro avg	0.92	0.92	0.92	1622
weighted avg	0.95	0.95	0.95	1622

Fig. 12. Classification report for Ensemble 2D CNN

Comparison with 3D CNN model: with the 3D CNN, the accuracy was 90.63% when trained in 50 epochs, indicating that the model correctly classified 91% of the test images. Thus, this may already be a pretty strong performance for a binary classification task in medical imaging, where clinically correct predictions are absolutely essential. Figure 14 and figure 15 shows the results and ROC curves of 3D CNN:

	precision	recall	f1-score	support
0	0.95	0.98	0.97	1340
1	0.91	0.78	0.84	282
accuracy			0.95	1622
macro avg	0.93	0.88	0.90	1622
weighted avg	0.95	0.95	0.95	1622

Fig. 13. Classification Report for 3D CNN

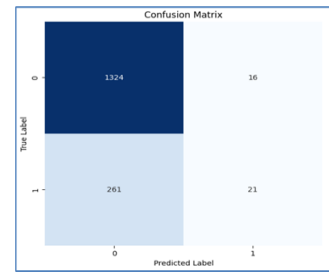


Fig. 14. Confusion matrix for 3D CNN

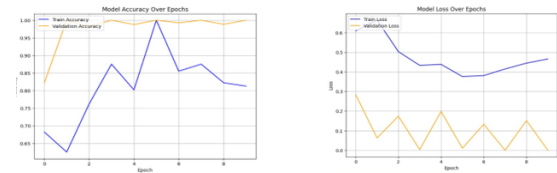


Fig. 15. Accuracy and Loss Curves for 3D CNN

TABLE I
COMPARISON WITH OTHER METHODOLOGIES

Methodology	Algorithms Used	Results
Taguchi Optimization with 2D CNN [7]	2D CNN, Taguchi Optimizer	94%
Segmentation Method [6]	CNN architecture using segmentation	73%
Adaptive Boosting Algorithm with CNN	CNN, AdaBoost Classifier	90%
2D CNN Ensemble Approach [16]	2D CNN architectures	95%
Proposed Method (2D CNN Ensemble)	Three 2D CNN architectures, Ensemble Approach	96%

IV. CONCLUSION

In this work, we propose an ensemble learning technique in the diagnosis of lung cancer via a CT scan image-based approach that relies on three different architectures of 2D CNNs. By using different depths and filter sizes in the creation of each CNN for capturing most of the properties of lung nodules, it ensures more reliability with respect to accuracy in classification. Compared to individual models, ensemble models worked better as classification accuracy improved to 96% by aggregating the predictions of several CNNs through soft voting. Further, it succeeded in handling class imbalance, particularly if Focal Loss is used and hard data to classify are targeted. This work demonstrates that an ensemble of deep learning methods can achieve superior accuracy and robustness in medical image analysis tasks, particularly for more complex applications like lung cancer diagnosis. Improvement in diagnostic performance in the ensemble model is then realized based on its ability to aggregate diverse feature extraction techniques applied differently by different CNNs, which also allows for a better comprehension of the input data.

V. LIMITATIONS AND FUTURE SCOPE

The major drawback is that the study makes use of 2D CNNs, which are not good at getting the 3D structure of lung nodules. Generally, lung nodules are three-dimensional; if one applies two-dimensional slices, they may lose important spatial information, which can propagate to affect the performance of the model. Future work in this field should include 3D CNN architectures that may very well increase classification accuracy far more effectively by analyzing volumetric data.

Future work may also explore the application of transfer learning. Transfer learning is accomplished through the employment of pre-trained networks that already learned informative features from large medical image collections, and might be applied for further improvement in model generalization performance on new datasets. The complementing properties of many different classifiers might be exploited for further improvement in model performance by exploring other ensemble methods, including stacking or boosting.

VI. DATASET REFERENCE

Setio, A.A.A., Traverso, A., de Bel, T., Berens, M.S. LUNA16: A challenge for automatic nodule detection in CT scans. Grand Challenge, available at: <https://luna16.grand-challenge.org/>.

REFERENCES

- [1] L. Wang, "Deep learning techniques to diagnose lung cancer," *Cancers*, vol. 14, no. 22, p. 5569, 2022. [Online]. Available: <https://doi.org/10.3390/cancers14225569>
- [2] M. A. Heuvelmans *et al.*, "Lung cancer prediction by deep learning to identify benign lung nodules," *Lung Cancer*, vol. 154, p. 14, 2021. [Online]. Available: <https://doi.org/10.1016/j.lungcan.2021.01.027>
- [3] Z. Li *et al.*, "Deep learning methods for lung cancer segmentation in whole-slide histopathology images—the ACDC@LUNGp challenge 2019," *IEEE J. Biomed. Health Inform.*, vol. 25, no. 2, pp. 429–440, 2020. [Online]. Available: <https://doi.org/10.1109/JBHI.2020.3039741>
- [4] P. Huang *et al.*, "Prediction of lung cancer risk at follow-up screening with low-dose CT: a training and validation study of a deep learning method," *The Lancet Digital Health*, vol. 1, no. 7, pp. e353–e362, 2019. [Online]. Available: <https://doi.org/10.1016/j.lungcan.2021.01.027>
- [5] R. R. Subramanian *et al.*, "Lung cancer prediction using deep learning framework," *Int. J. Control Autom.*, vol. 13, no. 3, pp. 154–160, 2020. [Online]. Available: <https://www.researchgate.net/publication/346700750>
- [6] A. Shimazaki *et al.*, "Deep learning-based algorithm for lung cancer detection on chest radiographs using the segmentation method," *Sci. Rep.*, vol. 12, no. 1, p. 727, 2022. [Online]. Available: <https://doi.org/10.1038/s41598-021-04667-w>
- [7] C. J. Lin, S. Y. Jeng, and M. K. Chen, "Using 2D CNN with Taguchi parametric optimization for lung cancer recognition from CT images," *Appl. Sci.*, vol. 10, no. 7, p. 2591, 2020. [Online]. Available: <https://doi.org/10.3390/app10025991>
- [8] J. Zhou *et al.*, "An ensemble deep learning model for risk stratification of invasive lung adenocarcinoma using thin-slice CT," *NPJ Digital Med.*, vol. 6, no. 1, p. 119, 2023. [Online]. Available: <https://doi.org/10.1038/s41746-023-00866-z>
- [9] R. Tekade and K. Rajeswari, "Lung cancer detection and classification using deep learning," in *Proc. 2018 Fourth Int. Conf. Comput. Commun. Control Autom. (ICCUBE)*, Pune, India, 2018, pp. 1–5. [Online]. Available: <https://doi.org/10.1109/ICCUBE.2018.8697352>
- [10] S. Vijn, P. Gaurav, and H. M. Pandey, "Hybrid bio-inspired algorithm and convolutional neural network for automatic lung tumor detection," *Neural Comput. Appl.*, vol. 35, no. 33, pp. 23711–23724, 2023. [Online]. Available: <https://doi.org/10.1007/s00521-020-05362-z>
- [11] S. Makaju *et al.*, "Lung cancer detection using CT scan images," *Procedia Comput. Sci.*, vol. 125, pp. 107–114, 2018. [Online]. Available: <https://doi.org/10.1016/j.procs.2017.12.016>
- [12] N. Kalaivani *et al.*, "Deep learning based lung cancer detection and classification," in *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 994, no. 1, p. 012026, 2020. [Online]. Available: <https://doi.org/10.1088/1757-899X/994/1/012026>
- [13] P. M. Shakeel *et al.*, "Lung cancer detection from CT image using improved profuse clustering and deep learning instantaneously trained neural networks," *Measurement*, vol. 145, pp. 702–712, 2019. [Online]. Available: <https://doi.org/10.1016/j.measurement.2019.05.027>
- [14] S. Moturi *et al.*, "Prediction of liver disease using machine learning algorithms," in *Proc. Int. Conf. Data Sci. Appl.*, Singapore, 2023, pp. 243–254. [Online]. Available: https://doi.org/10.1007/978-981-99-7817-5_19
- [15] A. Seva, S. T. Rao, and M. Sireesha, "Prediction of liver disease with random forest classifier through SMOTE-ENN balancing," in *2024 IEEE 13th Int. Conf. Commun. Syst. Netw. Technol. (CSNT)*, 2024, pp. 928–933. [Online]. Available: <https://doi.org/10.1109/CSNT60213.2024.10546170>

ORIGINALITY REPORT

7%

SIMILARITY INDEX

4%

INTERNET SOURCES

5%

PUBLICATIONS

0%

STUDENT PAPERS

PRIMARY SOURCES

1

jcreview.com

Internet Source

1%

2

www.aicte-india.org

Internet Source

1%

3

www.ncbi.nlm.nih.gov

Internet Source

1%

4

Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dharendra Kumar Shukla. "Artificial Intelligence, Blockchain, Computing and Security", CRC Press, 2023

Publication

1%

5

Zhang Li, Jiehua Zhang, Tao Tan, Xichao Teng et al. "Deep Learning Methods for Lung Cancer Segmentation in Whole-slide Histopathology Images - the ACDC@LungHP Challenge 2019", IEEE Journal of Biomedical and Health Informatics, 2020

Publication

<1%

6

Wanli Yang, Yimin Chen, Chen Huang, Mingke Gao. "Video-Based Human Action Recognition

<1%