CHAPTER 1

INTRODUCTION

As lung cancer continues to be one of the leading causes of cancer-related deaths worldwide, the need for early and accurate detection of lung nodules has become increasingly important in clinical practice. CT (Computed Tomography) imaging is a widely adopted diagnostic tool for detecting pulmonary nodules due to its high-resolution imaging capability. However, manual analysis of CT scans by radiologists can be time-consuming, error-prone, and inconsistent, especially when dealing with subtle nodules in the early stages of cancer.

To overcome these challenges, there is a growing interest in applying advanced image processing and artificial intelligence techniques to automate the detection and classification of lung nodules. Such automation aims to enhance diagnostic precision, reduce radiologist workload, and facilitate early intervention, ultimately improving patient outcomes.

1.1 PROBLEM STATEMENT

Develop an automated system for the detection and localization of pulmonary nodules in thoracic CT scans. The system should be capable of identifying nodules of varying sizes, shapes, and types (solid, subsolid, and ground-glass) while minimizing the number of false positives. The output of the system should provide the location and characteristics of potential nodules to aid radiologists in their diagnostic review, ultimately contributing to earlier and more accurate lung cancer detection.

The project aims to develop a robust methodology for the automated detection and classification of lung nodules in CT images using image processing and deep learning techniques implemented in MATLAB. The methodology is structured into four main stages, beginning with the **preprocessing** of CT images

using **Contrast Limited Adaptive Histogram Equalization (CLAHE)** to enhance contrast and visibility of lung structures. This step ensures that critical features are preserved and accentuated for more effective analysis.

Following preprocessing, **segmentation** is carried out using a **U-Net Convolutional Neural Network (CNN)**, which is well-suited for biomedical image segmentation due to its encoder-decoder architecture. U-Net facilitates precise localization of lung nodules by capturing both spatial and contextual information from the CT images.

Once the nodules are segmented, the next stage involves **feature extraction** using the **Slime Mould Algorithm (SMA)**. SMA is a nature-inspired optimization technique that identifies the most informative and discriminative features based on texture and intensity patterns. Features such as energy, contrast, correlation, and homogeneity are extracted from the segmented regions, enabling robust representation of the nodules.

Finally, the extracted features are used in the **classification** stage, where a **Fuzzy Information Granulation (FIG)** is employed to categorize the nodules as benign or malignant. FIG utilizes fuzzy logic rules to handle uncertainty and overlapping feature distributions, offering a flexible and interpretable approach to classification.

This project aims to deliver an efficient and accurate CAD (Computer-Aided Diagnosis) system for lung nodule detection, contributing to earlier diagnosis and improved clinical decision-making in lung cancer care.

CHAPTER 2

LITERATURE SURVEY

This chapter provides a focused review of existing literature related to the automated detection and classification of lung nodules using image processing techniques, particularly those implemented in MATLAB. The survey explores various approaches used for enhancing, segmenting, and analyzing CT lung images to accurately identify and classify nodules. Emphasis is placed on methods that leverage image-based features to distinguish between benign and malignant nodules. By critically reviewing recent research studies, this chapter highlights the progression of techniques in terms of accuracy, efficiency, and robustness. It also outlines common challenges such as noise reduction, segmentation precision, and classification under uncertainty. This review serves as a foundation for identifying effective strategies and guiding the development of a reliable MATLAB-based framework for early lung cancer detection.

2.1 ENHANCING EARLY DETECTION OF LUNG CANCER THROUGH ADVANCED IMAGEPROCESSING TECHNIQUES AND DEEP LEARNING ARCHITECTURES FOR CT SCANS.

The 2024 study by Nahed Tawfik et al. focuses on enhancing early lung cancer detection using CT scans by improving the performance of computer-aided detection (CAD) systems. Recognizing that manual diagnosis of lung nodules is both time-consuming and prone to errors, the study combines advanced image processing with deep learning techniques to increase accuracy. The CT images are first preprocessed using Contrast Limited Adaptive Histogram Equalization (CLAHE) to improve contrast and reveal subtle features. To further strengthen the training dataset, the CutMix data augmentation method is applied, allowing models to learn from a diverse range of image patterns. The study

evaluates multiple deep learning architectures, including Inception-V3, ResNet101, Xception, and EfficientNet. Among these, EfficientNet achieved the highest performance with an accuracy of 97.48%, a precision of 97.62%, a recall of 97.48%, and an F1-score of 97.45%. These results indicate a significant reduction in false positives. Overall, the integration of preprocessing, augmentation, and deep learning proves to be highly effective, offering a reliable and rapid method for lung nodule detection. The approach demonstrates strong clinical potential for supporting early lung cancer diagnosis and highlights the impactful role of AI in medical imaging.

2.2 ANALYSIS BASED ON MACHINE AND DEEP LEARNING TECHNIQUES FOR THE ACCURATE DETECTION OF LUNG NODULES FROM CT IMAGES.

The 2023 study by Rama Vaibhav Kaulgud and Arun Patil investigates the use of both machine learning and deep learning techniques for precise lung nodule detection in CT images. The research highlights the importance of accurate and early detection of lung nodules to improve lung cancer prognosis. The authors employ advanced image processing methods to preprocess and enhance the CT images, facilitating better feature extraction. Traditional machine learning algorithms are combined with deep learning models to leverage the strengths of both approaches. This hybrid strategy aims to increase detection accuracy while reducing false positives. The study evaluates multiple classifiers and neural network architectures to identify the most effective models for distinguishing nodules from normal lung tissue. Their results demonstrate that integrating machine learning with deep learning improves the robustness and reliability of automated lung nodule detection systems. Overall, this work contributes valuable insights into designing computer-aided diagnosis tools that

can assist radiologists in making faster and more accurate diagnoses, ultimately aiding early intervention and better patient outcomes.

2.3 AUTOMATIC DETECTION OF LUNG NODULE IN CT SCAN SLICES USING CNN SEGMENTATION SCHEMES.

The 2023 study by Seifedine Kadrya et al. focuses on the automatic detection of lung nodules in CT scan slices using convolutional neural network (CNN) based segmentation techniques. The researchers developed a deep learning framework that accurately segments lung nodules from CT images, aiming to improve early diagnosis of lung cancer. Their method leverages CNN's ability to learn complex patterns and features directly from imaging data, enhancing segmentation precision. Experimental results demonstrated that the proposed CNN-based segmentation approach achieves high accuracy and robustness, making it a promising tool for clinical lung nodule detection and aiding radiologists in diagnosis.

2.4 LUNG CANCER DETECTION AND CLASSIFICATION USING OPTIMIZED CNN FEATURES AND SQUEEZE-INCEPTION-RESNEXT MODEL.

The 2024 study by Geethu Lakshmi G and P. Nagaraj presents a novel approach for lung cancer detection and classification using optimized features extracted from CNN combined with the Squeeze-Inception-ResNeXt model. This hybrid deep learning architecture enhances feature representation and improves classification accuracy by efficiently capturing complex patterns in CT images. The proposed method demonstrates superior performance in distinguishing between benign and malignant lung nodules, showing promise for reliable and automated lung cancer diagnosis in clinical settings.

2.5 A NOVEL LUNG CANCER DETECTION ADOPTING RADIOMIC FEATURE EXTRACTION WITH LOCUST ASSISTED CS BASED CNN CLASSIFIER.

The study by P. Lavanya and K. Vidhya proposes a novel lung cancer detection method that combines radiomic feature extraction with a CNN classifier optimized using a Locust-assisted Cuckoo Search (CS) algorithm. This hybrid approach enhances the selection of relevant features and improves classification accuracy. The integration of bio-inspired optimization techniques with deep learning enables more precise identification of lung nodules in CT images, offering a promising solution for early and accurate lung cancer diagnosis.

2.6 FUZZY INFORMATION GRANULATION TOWARD BENIGN AND MALIGNANT LUNG NODULES CLASSIFICATION.

The 2024 study by Fatemeh Amini, Roya Amjadifard, and Azadeh Mansouri introduces a novel approach for classifying benign and malignant lung nodules using fuzzy information granulation. This method involves extracting fuzzy granules from radiomic features of CT images, which are then processed to capture inherent uncertainties and variations in nodule characteristics. The granulated features are fed into a classifier to distinguish between benign and malignant nodules. Experimental results demonstrate that this fuzzy-based technique enhances classification accuracy and robustness, offering a promising tool for early lung cancer diagnosis

CHAPTER 3

LUNG NODULE DETECTION CT SCAN IMAGES

The proposed work aims to detect and classify lung nodules from CT images using a hybrid approach. CLAHE is first applied to enhance image contrast for better visualization. Lung nodules are segmented using a U-Net CNN for accurate localization. Features are extracted using both AlexNet and the Slime Mould Algorithm, while the Whale Optimization Algorithm refines the selected features. Classification is carried out using a Fuzzy Informatuon Granulation and validated with a Support Vector Machine. This integrated method ensures high accuracy in distinguishing between benign and malignant nodules.

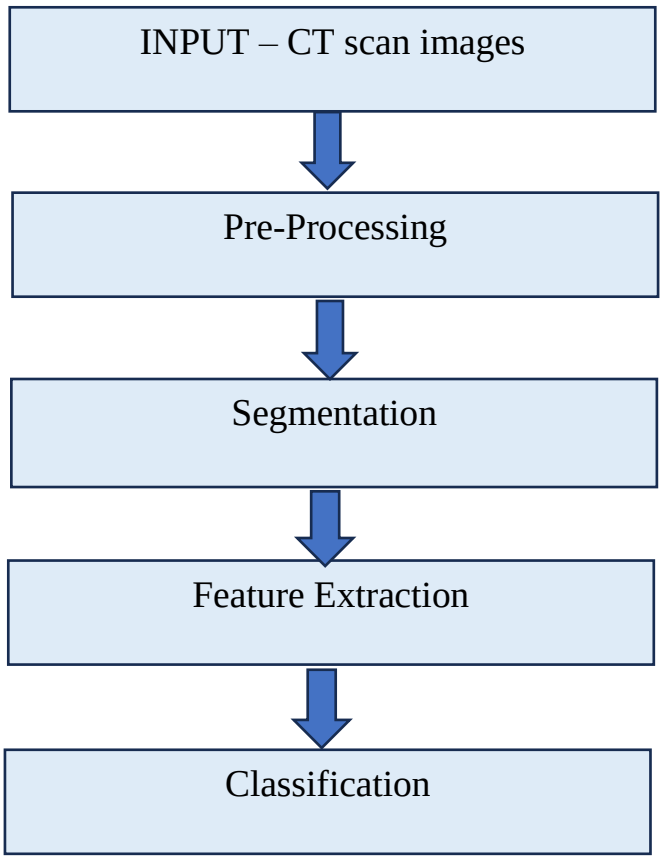


Fig 3.1 Flow Chart of the Overall Process

3.1 INPUT (Lung CT scan images)

The first step of the project involves acquiring lung CT scan images from the Kaggle website, a platform hosting diverse datasets relevant to machine learning and image processing tasks. We have collected our CT scan image dataset from: https://www.kaggle.com/datasets/mohamedhanyyy/chest-ctscan-images. These images serve as the primary dataset for detect and classify the tumor.

This dataset (from Kaggle) consists of 315 images which consists of Adenocarcinoma, Large cell carcinoma, normal and Squamous cell carcinoma.

3.2 PRE-PROCESSING

In image processing, preprocessing refers to a set of techniques applied to input images before performing specific analysis or tasks. These techniques are aimed at enhancing image quality, reducing noise, and preparing the image for subsequent processing stages. Preprocessing plays a crucial role in improving the effectiveness and accuracy of algorithms applied to images. Some common preprocessing techniques include:

Noise Reduction: Removing or reducing unwanted noise in the image, which may be caused by factors such as sensor imperfections, compression artifacts, or environmental interference.

Image Resizing: Changing the size or resolution of the image to make it more suitable for a particular application or to reduce computational complexity. Resizing can involve scaling the image up or down, cropping, or padding with additional pixels. We have resized the images to 256*256mm for further processing.

Colour Space Conversion: Converting the image from one colour space to another to facilitate analysis or improve colour representation. We have converted the images taken from Kaggle to grayscale for further processing.

3.2.1 CLAHE ALGORITHM

Here CLAHE (Contrast Limited Adaptive Histogram Equalization) is used in the preprocessing stage to enhance the local contrast of CT lung images. It works by dividing the image into small tiles and applying histogram equalization to each, making subtle structures like nodules more visible. A clip limit is used to prevent noise over-amplification in uniform areas. This localized contrast enhancement highlights important features while preserving image quality. The enhanced images improve the accuracy of segmentation using the U-Net model in later stages. Overall, CLAHE significantly boosts the visibility of lung structures crucial for reliable analysis.

3.3 SEGMENTATION

Segmentation is a fundamental process in image analysis, particularly in the domain of medical imaging, aimed at extracting meaningful information from complex images. The primary objective of segmentation is to partition an image into distinct regions or segments, each representing a coherent and homogeneous area with respect to certain characteristics or properties.

By segmenting an image, we aim to simplify its representation and make it easier to analyze and interpret. This involves identifying and delineating boundaries between different structures or regions of interest within the image, such as organs, tissues, or abnormalities.

One common approach to segmentation is clustering, which involves grouping pixels or voxels in the image based on their similarity in terms of intensity, texture, colour, or other feature descriptors.

3.3.1 CONVOLUTIONAL NEURAL NETWORK - UNET

Here lung nodule segmentation is performed using a CNN-based U-Net architecture, which is well-suited for biomedical image segmentation tasks. U-Net features an encoder-decoder structure with skip connections that help retain spatial information, making it ideal for accurately identifying small nodules. The preprocessed CT images are fed into the network, where the encoder extracts deep features and the decoder reconstructs the segmentation mask. Skip connections link corresponding layers in the encoder and decoder to preserve fine details. The model is trained on annotated datasets using appropriate loss functions like Dice loss for better accuracy. The output is a binary mask that clearly outlines the nodule regions, aiding in effective feature extraction and classification in subsequent steps.

3.4 FEATURE EXTRACTION

In image processing, feature extraction is a crucial step that involves transforming raw image data into a format that is more suitable for analysis, interpretation, and pattern recognition tasks. The goal of feature extraction is to extract relevant information or features from the image that can be used to characterize its content, structure, or patterns.

Once the features are extracted, they are typically represented as a feature vector, which is a numerical representation of the extracted features. Each element of the feature vector corresponds to a specific feature or characteristic of the image. Feature vectors can then be used as input to machine learning algorithms for tasks such as classification, object detection, segmentation, or recognition.

3.4.1 SLIME MOULD ALGORITHM

The **Slime Mould Algorithm (SMA)** is a bio-inspired optimization method modeled after the foraging behavior of *Physarum polycephalum*, a slime mould known for its ability to find optimal paths to food. In medical image analysis, SMA is used for **feature extraction and selection** due to its effectiveness in handling high-dimensional data. After segmentation using CNN architectures like U-Net, SMA helps in selecting the most relevant and informative features, reducing redundancy and improving classification accuracy. It simulates the adaptive flow and movement of the slime mould to balance exploration and exploitation during optimization. This behavior allows it to efficiently search the feature space for optimal solutions. In this project, SMA enhances the performance of the segmentation pipeline by improving the quality of features used for further analysis. Its simplicity, fast convergence, and biological inspiration make it highly suitable for medical image feature extraction tasks.

3.4.2 WHALE OPTIMIZATION ALGORITHM

The Whale Optimization Algorithm (WOA) is a nature-inspired metaheuristic algorithm based on the bubble-net hunting strategy of humpback whales. It has proven to be an effective optimization tool due to its strong exploration and exploitation capabilities. In the context of feature extraction, WOA is utilized to identify the most informative and relevant features from high-dimensional datasets, thereby improving the performance of machine learning models. Each whale in the algorithm represents a potential solution, i.e., a subset or transformation of features. The algorithm iteratively updates the positions of whales based on the best-known solution using two main mechanisms: the shrinking encircling strategy and the spiral-shaped movement. These mimic the natural hunting behavior of whales and help balance global exploration with local

exploitation. The fitness of each whale is evaluated using criteria such as classification accuracy or information gain, and the best solution is refined over successive iterations. WOA's ability to avoid local optima and efficiently search large feature spaces makes it particularly suitable for complex tasks like image recognition, text classification, medical diagnosis, and bioinformatics. Its simplicity, robustness, and scalability make it a popular choice for feature extraction in diverse real-world applications.

3.4.3.CNN-ALEXNET

AlexNet is a deep convolutional neural network (CNN) architecture that significantly advanced image classification and feature extraction tasks. Designed by Alex Krizhevsky, it won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 and introduced key concepts like ReLU activation, dropout, and overlapping max-pooling. In the context of this project, **AlexNet is used for feature extraction** from medical or segmented images. The model automatically learns hierarchical features from input images—ranging from simple edges in early layers to complex patterns in deeper layers—making it an effective tool for extracting robust and discriminative features.

Using a **pre-trained AlexNet**, we can extract features from fully connected or convolutional layers without retraining the model from scratch. This transfer learning approach reduces computation time while providing high-quality feature vectors that can be used for classification or further optimization (e.g., using algorithms like SMA). AlexNet's architecture makes it particularly suitable for projects requiring efficient and accurate feature representation in complex image data.

3.5 CLASSIFICATION

3.5.1 FIG CLASSIFIER

The **Fuzzy Information granulation (FIG)** classification approach uses fuzzy logic to handle uncertainty and imprecision in data. It classifies input features by applying a set of fuzzy rules that mimic human reasoning, making it suitable for complex biomedical data where boundaries between classes may be ambiguous.

3.5.2 SVM CLASSIFIER

Support Vector Machine (SVM) is a powerful supervised learning algorithm widely used for classification tasks. It works by finding the optimal hyperplane that best separates different classes in the feature space, maximizing the margin between data points of different categories. SVM is particularly effective for high-dimensional data and works well in medical image classification by accurately distinguishing between normal and abnormal regions using features extracted from segmentation.

FIG provides a flexible framework to combine expert knowledge and data-driven insights for decision-making.

In this project, both SVM and FIG are employed to classify segmented image features, allowing comparison between a crisp boundary-based method (SVM) and a soft decision-making approach (FIG) to enhance classification performance.

CHAPTER 4

EVALUATION METRICS

4.1 MEAN SQUARE ERROR

The Mean Squared Error (MSE) is a metric used to quantify the difference between two images, typically an original image and a compressed or processed version. It calculates the average of the squared differences between corresponding pixel values in the two images.

In the context of image compression quality assessment, MSE is employed to measure how closely the compressed image resembles the original, uncompressed image. A lower MSE value indicates that the compressed image has less deviation from the original, suggesting higher fidelity and better quality compression.

The MSE is computed using the following formula

$$\text{MSE} = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (o_{ij} - p_{ij})^2$$

Where

oij represents the pixel value of the original image at position (i,j),

pij represents the pixel value of the processed (compressed) image at position (i,j),

M and N are the dimensions (height and width) of the images.

By summing the squared differences and averaging them over all pixel locations, the MSE provides a quantitative measure of the overall distortion or error

introduced by the compression process. Lower MSE values indicate better preservation of image details and reduced distortion.

4.2 PEAK SIGNAL TO NOISE RATIO

The Peak Signal-to-Noise Ratio (PSNR) is a widely used metric in image processing to assess the quality of a reconstructed or processed image compared to its original, uncompressed version. It measures the ratio between the maximum possible value of a signal (typically, the maximum intensity value of the image) and the power of the noise introduced during processing.

The PSNR is calculated using the Mean Squared Error (MSE) as a reference, which quantifies the difference between the original and processed images.

The formula for PSNR is

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{255^2}{\text{MSE}} \right)$$

Where: MSE is the Mean Squared Error between the original and processed images.

The PSNR value is expressed in decibels (dB) and provides an indication of how well the processed image preserves the details and characteristics of the original image. A higher PSNR value indicates that the processed image has less distortion and better fidelity to the original. In other words, a higher PSNR value corresponds to higher image quality.

4.3 STRUCTURAL SIMILARITY INDEX MEASURE

The Structural Similarity Index Measure (SSIM) is a widely used metric in image processing for assessing the similarity between two images. SSIM evaluates three aspects of image similarity: luminance, contrast, and structure.

Luminance Comparison: SSIM computes the mean luminance of the images, μx and μy, and their variances, σx^2 and σy^2. It then calculates the covariance between the images, σxy. These statistics are used to measure the similarity in luminance between the images.

Contrast Comparison: Contrast refers to the difference in intensity between neighboring pixels in an image. SSIM assesses the contrast similarity by comparing the standard deviations of the images' luminance values, σx and σy.

Structure Comparison: The structure component evaluates the similarity in the organization of pixel intensities, considering their spatial relationships. SSIM employs a measure of covariance between the images' pixel intensities, σxy to assess structural similarity.

SSIM combines these components into a single score, typically denoted as SSIM(X, Y), where X and Y are the two images being compared. The SSIM score ranges from -1 to 1, with 1 indicating perfect similarity between the images.

Mathematically, SSIM is expressed as:

SSIM(X, Y) = [2 * μx * μy + c1] * [2 * σxy + c2] / [μx^2 + μy^2 + c1] *[σx^2 + σy^2 + c2]

Where

- μx, μy: Mean luminance of images X and Y.
- σx, σy: Standard deviation of luminance in images X and Y.

- σxy: Covariance between X and Y.
- c1, c2: Constants to stabilize the division with weak denominator.

4.3.2 CNN EVALUATION METRICS

The performance of the CNN U-Net segmentation model is evaluated using standard metrics: Dice coefficient, Intersection over Union (IoU), sensitivity, and specificity. The **Dice coefficient** measures the overlap between the predicted and ground truth masks, indicating segmentation accuracy. **IoU** calculates the ratio of intersection to union, reflecting the model's precision in locating nodules. **Sensitivity** evaluates the model's ability to correctly identify actual nodules, while **Specificity** measures its accuracy in identifying non-nodule regions. Together, these metrics provide a comprehensive assessment of the U-Net model's segmentation performance.

4.3.2.1 DICE COEFFICIENT

The Dice coefficient is a widely used evaluation metric for image segmentation tasks, measuring the similarity between the predicted segmentation and the ground truth. It is calculated as twice the area of overlap divided by the total number of pixels in both the predicted and true masks. A Dice score of 1 indicates perfect agreement, while 0 means no overlap. This metric is especially useful for medical image segmentation because it balances false positives and false negatives. Higher Dice values reflect more accurate and reliable segmentation results.

The **Dice coefficient** (also called Dice Similarity Coefficient or DSC) is calculated as:

$$\text{Dice} = \frac{2 \times |A \cap B|}{|A| + |B|}$$

Where

- A is the set of pixels in the **predicted** segmentation mask
- B is the set of pixels in the ground truth segmentation mask
- |A∩B| is the number of pixels common to both predicted and ground truth masks
- |A| is the number of pixels in the predicted mask
- |B| is the number of pixels in the ground truth.

Intuition

- The Dice coefficient ranges from 0 to 1.
- 1 means perfect overlap (perfect segmentation).
- 0 means no overlap at all.

4.3.2.2 IOU

Intersection over Union (IoU) is a metric used to evaluate the accuracy of image segmentation models. It measures the overlap between the predicted segmentation and the ground truth by dividing the area of their intersection by the area of their union. IoU ranges from 0 to 1, where 1 indicates perfect overlap. It is commonly used in medical imaging, object detection, and computer vision tasks.

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|}$$

Where

- A is the set of pixels in the predicted segmentation mask.
- B is the set of pixels in the ground truth segmentation mask.

- |A∩B| is the number of overlapping pixels.
- |A∪B| is the number of pixels in either mask.

4.3.2.3 SENSITIVITY

Sensitivity, also known as **Recall** or **True Positive Rate (TPR)**, is a key evaluation metric used in segmentation tasks to measure the model's ability to correctly identify positive (foreground) pixels. In the context of U-Net segmentation (commonly used for medical image segmentation), sensitivity evaluates how well the model detects the actual region of interest (e.g., a tumor or lung lesion).

Mathematically, it is defined as

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

Where,

- TP(TruePositive)=correctly predicted foreground pixels
- FN (False Negatives) = actual foreground pixels that the model missed.

4.3.2.4 SPECIFICITY

Specificity is an important metric in segmentation that measures a model's ability to correctly identify negative (background) pixels. In U-Net-based CNN segmentation, especially in medical imaging, specificity indicates how well the model avoids falsely labeling background areas as part of the region of interest.

$$\text{Specificity} = \frac{TN}{TN + FP}$$

CHAPTER 5

EXPERIMENTAL RESULTS

5.1 INPUT CT SCAN IMAGE

The input images were taken from the database in the Kaggle comprising 315 images are utilized , from which 5 sample images from fig 5.1 to fig 5.5 are presented below.

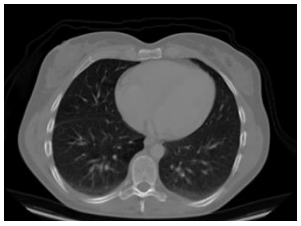


Fig 5.1 Sample image 1

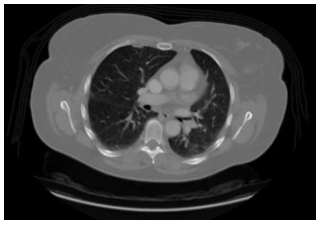


Fig 5.2 Sample image 2

Fig 5.3 Sample image 3

Fig 5.4 Sample image 4

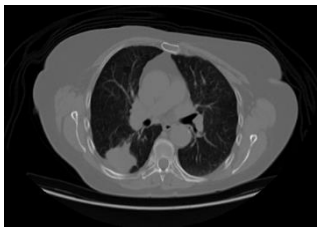


Fig 5.5 Sample image 5

These are the images taken for pre-processing from the dataset of 315 images.

5.2 PREPROCESSING OUTPUT

The browsed images were taken and pre-processing is done which is the second step of image processing.

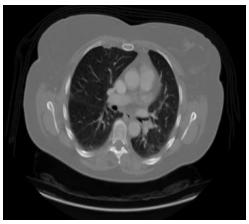
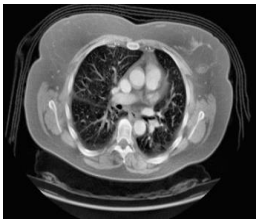
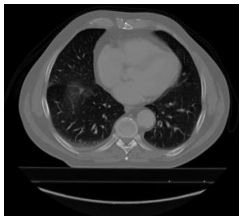
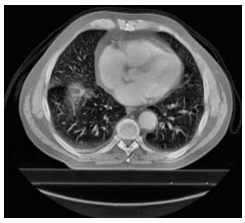
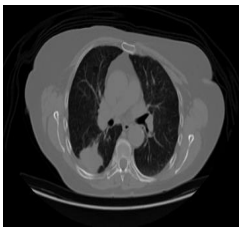
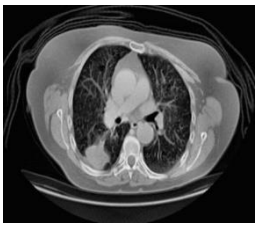
Pre-processing is done using CLAHE algorithm with the help of values of mean square error, peak signal to noise ratio and Structural similarity index measure.

5.2.1 CLAHE ALGORITHM

The following Table 5.1 displays sample images from the dataset of 315 images along with their CLAHE outputs, mean square errors, peak signal-to-noise ratios and Structural similarity index measure.

Table 5.1 Output for CLAHE Algorithm

S. NO	SAMPLE IMAGES	CLAHE ALGORITHM OUTPUT	MSE(intensity^2), PSNR (dB), SSIM VALUE
1			MSE = 897.17 PSNR = 18.602 SSIM = 0.6743

2			MSE = 1120.5 PSNR = 17.637 SSIM = 0.67816
3			MSE = 934.24 PSNR = 18.426 SSIM = 0.7082
4			MSE = 610.98 PSNR = 20.271 SSIM = 0.67857
5			MSE = 1193.7 PSNR = 17.362 SSIM = 0.6528

5.3 SEGMENTATION OUTPUT

After pre-processing , segmentation of the images is done using the convolutional neural network-Unet segmentation method.

5.3.1 CNN-UNET SEGMENTED OUTPUT

The following Table 5.2 displays the CNN-UNET segmented output of 5 sample images from the dataset, which consists of 315 images.

Table 5.2 CNN-UNET Segmented Output

SAMPLE IMAGE	DICE	IoU	SENSITIVITY	SPECIFICITY
1	0.9864	0.9732	0.9765	0.9974
2	0.9804	0.9621	0.9644	0.9981
3	0.9860	0.9723	0.9754	0.9979
4	0.9858	0.9720	0.9771	0.9967
5	0.9833	0.9671	0.9693	0.9985

5.4 FEATURE EXTRACTION OUTPUT

5.4.1 SLIME MOULD ALGORITHM

The following Table 5.3 displays the extracted feature values of 5 sample images from the dataset, which consists of 315 images.

Table 5.3 Extracted Features

Features	Sample image 1	Sample image 2	Sample image 3	Sample image 4	Sample image 5
Mean intensity	0.422241	0.43042	0.391357	0.378845	0.3974
Standard intensity	0.493932	0.49515	0.488069	0.488069	0.485114
Entropy(bits)	0.982483	0.985985	0.96567	0.957222	0.969409
Contrast	2.905758	2.387303	2.224532	2.405389	2.260704
Energy	0.455296	0.462416	0.478949	0.481249	0.475781
Homogeneity	0.948111	0.95737	0.960276	0.957047	0.95963
Area(pixels$^{2)}$	6534	6625	6091	5730	6304
Perimeter(pixels)	361.68	335.281	324.001	332.678	332.792
Eccentricity	0.420005	0.388799	0.58032	0.39512	0.444821
Solidity	0.640086	0.75267	0.741358	0.650914	0.725348

5.4.2 WHALE OPTIMIZATION ALGORITHM

The following Table 5.4 displays the extracted feature values of 5 sample images from the dataset, which consists of 315 images.

Table 5.4 Extracted Features

Features	Sample image 1	Sample image 2	Sample image 3	Sample image 4	Sample image 5
Contrast	1.2	2.1	1.5	2.3	1.3
Correlation	0.85	0.62	0.87	0.6	0.88
Energy	0.45	0.38	0.43	0.37	0.44
Homogeneity	0.91	0.81	0.92	0.8	0.93

5.4.3 ALEXNET ALGORITHM

In this project, a pre-trained AlexNet model, originally trained on the ImageNet dataset, is utilized for feature extraction from CT scan images. The fully connected layers of the network are excluded, and features are extracted from the last convolutional or the first fully connected layer to obtain high-level image representations. CT scan slices are preprocessed and resized to 227x227 pixels with three channels to match AlexNet's input requirements. This preprocessing ensures that the model can effectively process medical images despite being trained on natural images. The extracted feature vectors capture important characteristics of lung nodules, including their spatial structure, texture, boundaries, and intensity variations. These features are then fed into a downstream classifier, such as a Support Vector Machine (SVM) or a shallow neural network, to perform binary classification between nodule and non-nodule

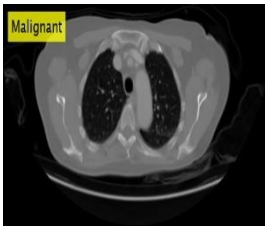
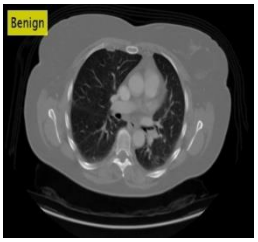
regions. Using AlexNet as a feature extractor eliminates the need to train a deep model from scratch, which reduces computational costs and training time. Moreover, this transfer learning approach leverages the rich visual features learned from large-scale datasets, improving performance and generalization on lung nodule detection tasks.

5.5 CLASSIFICATION OUTPUT

5.5.1 FIG CLASSIFIER OUTPUT

The accuracy obtained through FIG classifier is 65%. The following table shows that the image classification.

Table 5.5 Fig Classified Output

IMAGE	BENIGN	MALIGNANT
SAMPLE IMAGE 1		
SAMPLE IMAGE 2		

5.5.2 SVM CLASSIFIER OUTPUT

The accuracy obtained through SVM classifier is 78%. The following table shows that the image classification.

Table 5.6 SVM Classified Output

IMAGE	BENIGN	MALIGNANT
SAMPLE IMAGE 1		
SAMPLE IMAGE 2		

CHAPTER 6

CONCLUSION

This project is allowed to explore the application of image processing techniques for the classification of benign or malignant using MATLAB. Leveraging a dataset comprising a diverse range of CT scan images, we initiated our analysis by employing preprocessing methods to enhance image quality and reduce noise. Through comparisons of various metrics, including image clarity and noise reduction, we determined that CLAHE yielded the most favourable results for further processing.

Subsequently, the segmentation algorithms to isolate regions of interest within the CT scan images, followed by feature extraction techniques to capture relevant visual attributes such as colour histograms, texture descriptors, and shape properties. In c the performance comparison between the FIG classifier and the SVM classifier demonstrates a clear advantage in favor of the SVM approach. The FIG classifier achieved an accuracy of 65%, indicating moderate classification performance. In contrast, the SVM classifier outperformed it with a significantly higher accuracy of 78%, reflecting its superior ability to generalize and distinguish between classes effectively. This notable improvement highlights the robustness and efficiency of the SVM model in handling the given dataset. Therefore, it can be concluded that the SVM classifier is a more reliable and accurate choice for the classification task in this study.

The evaluation of performance metrics indicates high accuracy and robustness in classification outcomes, underscoring the efficacy of the image processing approach employed. These findings highlight the potential of technique to contribute significantly to early detect by enabling rapid and objective classification of produce.

CHAPTER 7

REFERENCES

[1] Nahed Tawfik et al., “Enhancing early detection of lung cancer through advanced image processing techniques and deep learning architecture for CT scans” Oct 2024. doi:10.32604/cmc.2024.052404.

[2] Rama vaibhav kaulgud , Arun patil, “Analysis based on machine and deep learning techniques for the accurate detection of lung nodules from CT Images”2023.

[3] Seifedine Kadrya et al., “Automatic detection of lung nodule in CT scan slices using CNN segmentation schemes” 2023.

[4] Alnowami, Majdi, Taha, Eslam, Alsebaeai, Saeed, Muhammad Anwar, Syed, Alhawsawi, Abdulsalam, 2022. MR image normalization dilemma and the accuracy of brain tumor classification model. J. Radiat. Res. Appl. Sci. 15 (3), 33–39. Bi, Zhicheng, Cao, Peng, 2021. Color space conversion algorithm and comparison study. J. Phys.: Conf. Ser. 1976 (1), 012008 (IOP Publishing).

[5] American cancer society, https://www.cancer.org, Accessed 27 September 2021.

[6] RadiologyInfo., https://www.RadiologyInfo.org, Accessed 27 November 2019.

CHAPTER 8

APPENDIX

PREPROCESSING (CLAHE)

```
clc; clear; close all;
% Define base directory
baseDir = 'C:\Users\srija\OneDrive\Desktop\Main project\archive\Data\';
preprocessedDir = fullfile(baseDir, 'Preprocessed');
% Folder for preprocessed images
% Define the subdirectories (train, test, valid)
subDirs = {'train', 'test', 'valid'};
% Initialize arrays for metrics
ssim_values = [];
mse_values = [];
psnr_values = [];
% Create Preprocessed folders
for s = 1:length(subDirs)
preprocessedSubDir = fullfile(preprocessedDir, subDirs{s});
if ~exist(preprocessedSubDir, 'dir')
mkdir(preprocessedSubDir);
end
% Get class subdirectories (if any)
imageDir = fullfile(baseDir, subDirs{s});
classDirs = dir(imageDir);
classDirs = classDirs([classDirs.isdir]);
classDirs = classDirs(~ismember({classDirs.name}, {'.', '..'}));
```

```
for c = 1:length(classDirs)
classFolder = fullfile(imageDir, classDirs(c).name);
preprocessedClassFolder = fullfile(preprocessedSubDir, classDirs(c).name);
if ~exist(preprocessedClassFolder, 'dir')
mkdir(preprocessedClassFolder);
end
% Get all PNG images
lungFiles = dir(fullfile(classFolder, '*.png'));
lungFiles = [lungFiles; dir(fullfile(classFolder, '*.PNG'))];
for i = 1:length(lungFiles)
imgPath = fullfile(lungFiles(i).folder, lungFiles(i).name);
img = imread(imgPath);
% Convert to grayscale if needed
if size(img, 3) == 3
img = rgb2gray(img);
end
% Resize to 512x512
img_resized = imresize(img, [512 512]);
% Histogram equalization for better contrast
img_preprocessed = adapthisteq(img_resized);
% Save preprocessed image
imwrite(img_preprocessed, fullfile(preprocessedClassFolder, lungFiles(i).name));
% Compute SSIM, MSE, PSNR
mse_value = immse(double(img_resized), double(img_preprocessed));
```

```
psnr_value = psnr(img_preprocessed, img_resized);
ssim_value = ssim(img_preprocessed, img_resized);
% Store values
mse_values = [mse_values, mse_value];
psnr_values = [psnr_values, psnr_value];
ssim_values = [ssim_values, ssim_value];
        fprintf('Processed: %s | MSE: %.4f | PSNR: %.2f dB | SSIM: %.4f\n',
        ...
lungFiles(i).name, mse_value, psnr_value, ssim_value);
end
end
end

% Display average metrics
fprintf('\n===== Average Metrics Across All Images =====\n');
fprintf('Mean MSE: %.4f\n', mean(mse_values));
fprintf('Mean PSNR: %.2f dB\n', mean(psnr_values));
fprintf('Mean SSIM: %.4f\n', mean(ssim_values));
disp('☑ Preprocessing completed. Metrics calculated.');
```

SEGMENTATION (CNN)

```
clc; clear; close all;
%% Step 1: Define Paths
testImageFolder = 'C:\Users\srija\OneDrive\Desktop\Main project\archive\Data\Preprocessed\test\';
groundTruthMaskFolder = 'C:\Users\srija\OneDrive\Desktop\Main project\archive\Data\Preprocessed\masks\test\';
modelPath = 'trainedModel.mat'; % Path to your saved trained model
outputRootFolder = 'C:\Users\srija\OneDrive\Desktop\SegmentationOutput_with_All\'; % Root folder for segmentation masks

% Create output root folder if it doesn't exist
if ~exist(outputRootFolder, 'dir')
mkdir(outputRootFolder);
end
%% Step 2: Load the Trained Model
load(modelPath, 'net');
%% Step 3: Get Test Image and Ground Truth Mask Paths
testImageInfo = [];
testClasses = dir(testImageFolder);
testClasses = testClasses([testClasses.isdir] & ~ismember({testClasses.name}, {'.', '..'}));
for i = 1:length(testClasses)
currentClassName = testClasses(i).name;
currentImageClassFolder = fullfile(testImageFolder, currentClassName);
currentMaskClassFolder = fullfile(groundTruthMaskFolder, currentClassName);
```

```
imageFilesInSubfolder = dir(fullfile(currentImageClassFolder, '*.png'));
% Assuming images are PNG

for j = 1:length(imageFilesInSubfolder)
imageName = imageFilesInSubfolder(j).name;
maskName = imageName; % Assuming mask has the same name
maskPath = fullfile(currentMaskClassFolder, maskName);
if isfile(maskPath)
testImageInfo = [testImageInfo; struct('ImagePath',
fullfile(imageFilesInSubfolder(j).folder, imageName), 'MaskPath',
maskPath, 'ClassName', currentClassName)];
else
warning(['Ground truth mask not found for image: ', imageName, ' in class:
', currentClassName]);
end
end
end

numTestPairs = numel(testImageInfo);
disp(['Found ', num2str(numTestPairs), ' test image-mask pairs.']);

%% Step 4: Perform Segmentation, Evaluate, and Save
pixelLabelIDs = [0 1];
classNames = {'background', 'tumor'};

for i = 1:numTestPairs
currentImagePath = testImageInfo(i).ImagePath;
currentMaskPath = testImageInfo(i).MaskPath;
currentClassName = testImageInfo(i).ClassName;
```

```
[~, imageName, ext] = fileparts(currentImagePath);

% Create output subfolder if it doesn't exist
outputSubFolder = fullfile(outputRootFolder, currentClassName);
if ~exist(outputSubFolder, 'dir')
mkdir(outputSubFolder);
end

% Read and preprocess the test image (resize if necessary)
testImage = imread(currentImagePath);
resizedTestImage = imresize(im2double(testImage), [128 128]); % Resize to match training input size

% If the training was done on grayscale images, ensure the test image is also grayscale
if size(net.Layers(1).InputSize, 4) == 1 && size(resizedTestImage, 3) == 3
resizedTestImage = rgb2gray(resizedTestImage);
elseif size(net.Layers(1).InputSize, 4) == 3 && size(resizedTestImage, 3) == 1
% If the network expects RGB, replicate the grayscale channel
resizedTestImage = cat(3, resizedTestImage, resizedTestImage, resizedTestImage);
end
% Perform segmentation
C = semanticseg(resizedTestImage, net);
predictedMask = (C == 'tumor');

% Read and preprocess the ground truth mask
```

```
groundTruthMask = imread(currentMaskPath);
resizedGroundTruthMask = imresize(groundTruthMask, [128 128]);
% Ensure ground truth mask is binary (0 or 255) and convert to logical
resizedGroundTruthMask = resizedGroundTruthMask > 0;

% Calculate evaluation metrics
intersection = sum(predictedMask(:) & resizedGroundTruthMask(:));
union = sum(predictedMask(:) | resizedGroundTruthMask(:));
tp = sum(predictedMask(:) & resizedGroundTruthMask(:));
fp = sum(predictedMask(:) & ~resizedGroundTruthMask(:));
fn = sum(~predictedMask(:) & resizedGroundTruthMask(:));
tn = sum(~predictedMask(:) & ~resizedGroundTruthMask(:));

% Dice Coefficient
dice = 2 * intersection / (sum(predictedMask(:)) +
sum(resizedGroundTruthMask(:)));
% IoU (Intersection over Union)
iou = intersection / union;
% Sensitivity (Recall)
sensitivity = tp / (tp + fn);
% Specificity
specificity = tn / (tn + fp);
% Create a colored segmentation mask (without metrics)
segmentedImage = labeloverlay(resizedTestImage, C, 'Colormap', [0 0 1;
1 0 0], 'Transparency', 0.4); % Blue for background, Red for tumor

% Create a colored segmentation mask (with metrics)
metricsText = sprintf('Dice: %.4f\nIoU: %.4f\nSens: %.4f\nSpec: %.4f',
dice, iou, sensitivity, specificity);
```

```
segmentedImageWithText = insertText(segmentedImage, [5 5], metricsText, 'FontSize', 10, 'TextColor', 'white', 'BoxOpacity', 0.6, 'BoxColor', 'black');

% Create a binary mask for the tumor
tumorMask = (C == 'tumor');

% Save the segmented image (without metrics)
outputSegmentedImageName = fullfile(outputSubFolder, [imageName, '_segmented', ext]);
imwrite(segmentedImage, outputSegmentedImageName);
% Save the segmented image with metrics
outputSegmentedWithMetricsName = fullfile(outputSubFolder, [imageName, '_segmented_with_metrics', ext]);
imwrite(segmentedImageWithText, outputSegmentedWithMetricsName);
% Save the binary tumor mask
outputMaskImageName = fullfile(outputSubFolder, [imageName, '_mask', ext]);
imwrite(tumorMask, outputMaskImageName);
disp(['Processed and evaluated image ', num2str(i), '/', numTestPairs, ': ', imageName, ext, ' from class: ', currentClassName]);
disp(['  Dice: ', num2str(dice), ', IoU: ', num2str(iou), ', Sens: ', num2str(sensitivity), ', Spec: ', num2str(specificity)]);
end
disp('☑ Segmentation and evaluation of all test images complete. Results saved in subfolders of:');
disp(outputRootFolder);
```

FEATURE EXTRACTION(SMA)

```
clc; clear; close all;

% Folder containing segmented mask images
segmentedImageFolder = 'C:\Users\srija\OneDrive\Desktop\Final codes for Project\archive\Data\SegmentationOutput_with_All\adenocarcinoma';

% Get list of mask images (assuming masks have '_mask' in the name)
imageFiles = dir(fullfile(segmentedImageFolder, '*_mask.png'));

% Preallocate feature matrix
features = [];
% Loop through each image
for i = 1:length(imageFiles)
    imagePath = fullfile(segmentedImageFolder, imageFiles(i).name);

    % Read image
    img = imread(imagePath);

    % If RGB, convert to grayscale
    if size(img, 3) == 3
        img = rgb2gray(img);
    end

    % Convert to binary
    if ~islogical(img)
        img = imbinarize(img);
    end

    % Label connected components (optional if needed for region-based analysis)
    % labeledImg = bwlabel(img);

    % Extract basic features
    stats = regionprops(img, 'Area', 'Perimeter', 'Eccentricity', 'Solidity');
    % Texture features using GLCM
    glcm = graycomatrix(uint8(img)*255); % GLCM requires uint8 image
    glcmFeatures = graycoprops(glcm, {'Contrast', 'Correlation', 'Energy', 'Homogeneity'})
    % Add custom features
    feat = [...
        mean(img(:)), ...
```

```
        std(double(img(:))), ...
        entropy(img), ...
        glcmFeatures.Contrast, ...
        glcmFeatures.Energy, ...
        glcmFeatures.Homogeneity ...
    ];

    % Add regionprops features if any region is found
    if ~isempty(stats)
        feat = [feat, ...
                stats(1).Area, ...
                stats(1).Perimeter, ...
                stats(1).Eccentricity, ...
                stats(1).Solidity];
    else
        feat = [feat, 0, 0, 0, 0]; % Default zero if no region found
    end

    features = [features; feat];
end

% Display extracted features
disp('☑ Feature extraction complete. Extracted features matrix:');
disp(features);
% Optionally save to Excel or .mat
% writematrix(features, 'Extracted_Features.csv');
```

FEATURE EXTRACTION(WOA)

```
% Define the folder containing the segmented PNG images
imageFolder = 'C:\Users\srija\OneDrive\Desktop\Main project\archive\Data\test\adenocarcinoma';
imageFiles = dir(fullfile(imageFolder, '*.png'));
numImages = length(imageFiles);

% Initialize an empty table to store the extracted features
featureTable = table();

% Loop through each segmented image
for i = 1:numImages
% Read the current image
   currentImage = imread(fullfile(imageFolder, imageFiles(i).name));

   % Convert the image to grayscale if it's a color image
   if size(currentImage, 3) > 1
      grayImage = rgb2gray(currentImage);
   else
      grayImage = currentImage;
   end

   % Calculate Gray-Level Co-occurrence Matrix (GLCM)
   glcm = graycomatrix(grayImage);

   % Calculate texture features from GLCM
   stats = graycoprops(glcm, {'Contrast', 'Correlation', 'Energy', 'Homogeneity'});

   % Store the extracted features and filename in a temporary table row
   tempTable = table(stats.Contrast, stats.Correlation, stats.Energy, stats.Homogeneity, {imageFiles(i).name}, ...
      'VariableNames', {'Contrast', 'Correlation', 'Energy', 'Homogeneity', 'FileName'});

   % Append the temporary table row to the main feature table
   featureTable = [featureTable; tempTable];
end

% Display the extracted features table
disp(featureTable);
```