VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018, KARNATAKA

A Project Report

on

LUNA - LUNG NODULE ANALYSIS

Submitted in partial fulfillment of the requirements for the VIII Semester of degree of
Bachelor of Engineering in Information Science and Engineering *of*
VisvesvarayaTechnological University, Belagavi

by

Apoorva Kashi	**1RN18IS023**	**Mohammed Misran**	**1RN18IS067**
Kirtana Sridharan	**1RN18IS061**	**Nikitha S**	**1RN18IS070**

Under the Guidance of

Dr. R Rajkumar

Associate Professor

Department of ISE

Department of Information Science and Engineering

RNS INSTITUTE OF TECHNOLOGY

Dr. Vishnuvardhan Road, Rajarajeshwari Nagar Post,
Channasandra, Bengaluru – 560 098

2021-2022

RNS INSTITUTE OF TECHNOLOGY

Dr. Vishnuvardhan Road, Rajarajeshwari Nagar Post
Channasandra, Bengaluru – 560 098

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the project work entitled ***LUNA - Lung Nodule Analysis*** has been successfully completed by **Apoorva Kashi (1RN18IS023), Kirtana Sridharan (1RN18IS061), Mohammed Misran (1RN18IS067), and Nikitha S (1RN18IS070),** bonafide students of **RNS Institute of Technology, Bengaluru** in partial fulfillment of the requirements for the award of degree **Bachelor of Engineering** in **Information Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during academic year **2021-2022**. The project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

Dr. R Rajkumar
Project Guide

Dr. Prakasha S / Mrs. Kusuma S
Project Coordinator

Dr. Suresh L
Professor and HOD

Dr. M K Venkatesha
Principal

External Viva

Name of the Examiners	**Signature with Date**
1. _____	**1.** _____
2. _____	**2.** _____

DECLARATION

We, **APOORVA KASHI [USN: 1RN18IS023]**, **KIRTANA SRIDHARAN [USN: 1RN18IS061], MOHAMMED MISRAN [USN: 1RN18IS067], NIKITHA S [USN: 1RN18IS070]** students of VIII Semester B.E. in Information Science and Engineering, RNS Institute of Technology hereby declare that the Project entitled ***LUNA- Lung Nodule Analysis*** has been carried out by us and submitted in partial fulfillment of the requirements for the *VIII Semester of degree of* ***Bachelor of Engineering in Information Science and Engineering*** *of Visvesvaraya Technological University, Belagavi* during academic year 2021-2022.

Place: Bangalore

Date:

APOORVA KASHI **(1RN18IS023)**

KIRTANA SRIDHARAN **(1RN18IS061)**

MOHAMMED MISRAN **(1RN18IS067)**

NIKITHA S **(1RN18IS070)**

ABSTRACT

Lung cancer is one of the most threatening diseases among all other lung disorders which is caused for uncontrolled cell growth. It is also responsible for more deaths per year than breast, prostate and colon cancer combined. The detection of lung cancer in early stages is the main comprehensible approach to enhance patient's survival rate. For medical imaging, many different types of images are used but Computer Tomography (CT) scans are generally preferred because of less noise. Image Processing together with machine learning process and other technologies are used to study medical images for earlier detection and treatment of present clinical world. Deep Learning is proven to be the best method for medical imaging, feature extraction and classification of objects. Several types of deep learning architectures are introduced by many researchers to classify the lung cancer.

This project aims to automate the lung cancer detection process where CT images are used to identify lung cancer at its early stage. The approach employed in the project for detecting cancerous nodules will have five rough steps: data loading, segmentation, grouping, classification, and nodule analysis and diagnosis. The CT scan images are segmented using U-Net architecture. The LUng Nodule Analysis (LUNA) Grand Challenge data is used to train the Convolutional Neural Network used for classification. The LUNA data contains CT scans, as well as human-annotated outputs for classification and grouping. Having high-quality data has a major impact on a project's success.

ACKNOWLEDGMENT

The fulfillment and rapture that go with the fruitful finishing of any assignment would be inadequate without the specifying the people who made it conceivable, whose steady direction and support delegated the endeavors with success.

We would like to profoundly thank **Management** of **RNS Institute of Technology** for providing such a healthy and vibrant environment to carry out the project work.

We would like to express our sincere thanks to our beloved Principal **Dr. M K Venkatesha** for his support and inspired me towards the attainment of knowledge.

We wish to place on record our words of gratitude to **Dr. Suresh L,** Professor and Head of the Department, Information Science and Engineering, for being the enzyme and master mind behind our project work.

We would also like to thank our project guide **Dr. R Rajkumar,** Associate Professor, Department of ISE, RNSIT, Bengaluru, for his valuable inputs, suggestions, time and guidance.

We place our thanks to project coordinators **Dr. Prakasha S**, Associate Professor and **Mrs. Kusuma S**, Assistant Professor, ISE, RNSIT for their timely guidelines and suggestions for carrying out the project work successfully.

We would like to thank all other teaching and non-teaching staff of Information Science & Engineering who have directly or indirectly helped me to carry out the project work.

Place: Bengaluru
Date:

APOORVA KASHI
KIRTANA SRIDHARAN
MOHAMMED MISRAN
NIKITHA S

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AUC	Area Under Curve
CAD	Computer Aided Design
CNN	Convolutional Neural Network
CT	Computed Tomography
DCNN	Deep Convolutional Neural Network
DICOM	Digital Imaging and Communications in Medicine
DSB	Data Science Bowl
GLCM	Gray-Level Co-occurrence Matrix
GPU	Graphical Processing Unit
IDRI	Image Database Resource Initiative
LIDC	Lung Image Database Consortium
MRI	Magnetic Resonance Imaging
NLST	National Lung Screening Trial
R-CNN	Recurrent Convolutional Neural Network
ROC	Receiver Operating Characteristic
RPN	Region Proposal Network
RRCNN	Recurrent Residual Convolutional Neural Network
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine

Chapter 1

INTRODUCTION

1.1 Background

Lung Cancer is an infectious lung tumor caused by uncontrollable tissue growth in the lungs. Cancer is one of the most serious and widespread disease that is responsible for large number of deaths every year. The uncontrollable division of undesirable cells in the lung region can be classified as Lung Cancer. Lung cancer disease is the second largest death threat to the world after heart attack, as this cancer is responsible for the largest number of deaths, compared to the number of deaths caused by any other cancer type. The main cause of such high death rate is the detection in later stages. Since there are no apparent signs or symptoms of early lung cancer, the clinical diagnosis of lung cancers are often late, making treatment expensive and ineffective. If lung cancer is detected at an earlier stage, chances of survival can increase up to 50-70% from 15%.

There are many techniques to diagnose the lung cancer such as X-rays, Computed Tomography (CT), Magnetic Resonance Imaging (MRI scan), and Sputum Cytology. The problem with these techniques is that it can be time consuming and makes detection possible only at later stages.

1.1.1 What is a CT scan?

CT scan or computed tomography scan (formerly known as computed axial tomography or CAT scan) is a medical imaging technique used in radiology to obtain detailed internal images of the body noninvasively for diagnostic purposes. The personnel that perform CT scans are called radiographers or radiology technologists. CT scanners use a rotating X-ray tube and a row of detectors placed in the gantry to measure X-ray attenuations by different tissues inside the body. CT scans are essentially 3D X-rays, represented as a 3D array of single-channel data. CT scans actually measure radio density, which is a function of both mass density and atomic number of the material under examination. For this project, the distinction isn't relevant, since the model will consume and learn from the CT data no matter what the exact units of the input happen to be. The volumetric thoracic CT is a common imaging tool for lung cancer diagnosis. It visualizes all tissues according to their absorption of X-ray. The lesion in the lung is called pulmonary nodules. A nodule usually has the same absorption level as the normal tissues but has a distinctive shape: the bronchus

and vessels are continuous pipe systems, thick at the root and thin at the branch, and nodules are usually spherical and isolated. It usually takes an experienced doctor around 10 minutes to perform a thorough check for a patient, because some nodules are small and hard to be found. The primary difference between CT scans and X-rays is that whereas an X-ray is a projection of 3D intensity (in this case, tissue and bone density) onto a 2D plane, a CT scan retains the third dimension of the data. This allows the user to render the data in a variety of ways: for example, as a grayscale solid.

1.1.2 Voxel

A voxel is the 3D equivalent to the familiar two-dimensional pixel. It encloses a volume of space (hence, "volumetric pixel"), rather than an area, and is typically arranged in a 3D grid to represent a field of data. Each of those dimensions will have a measurable distance associated with it. Often, voxels are cubic, but the project will be dealing with voxels that are rectangular prisms. Each voxel of a CT scan has a numeric value that roughly corresponds to the average mass density of the matter contained inside. Most visualizations of that data show high-density material like bones and metal implants as white, low-density air and lung tissue as black, and fat and tissue as various shades of gray.

1.2 Existing Systems and their Limitations

Existing system makes use of NumPy arrays for cancer detection but lack in the ability to perform very fast operations on graphical processing units (GPUs), distribute operations on multiple devices or machines, and keep track of the graph of computations that created them. These are all important features when implementing a modern deep learning library.

Segmentation methods in existing system uses instance segmentation which labels individual objects of interest with distinct labels. Whereas semantic segmentation would label a picture of two people with two labels ("person" and "background"), instance segmentation would have three labels ("person1," "person2," and "background") with a boundary somewhere around the labels. While this could be useful to distinguish "nodule1" from "nodule2," instead grouping is used to identify individual nodules. Another approach to these kinds of tasks is object detection, which locates an item of interest in an image and puts a bounding box around the item. While both instance segmentation and object detection could be great use, their implementations are somewhat complex. Also, training object-detection models typically requires much more computational resources than the approach requires.

1.3 Proposed System

In the first step raw CT scan data are loaded into a form that can be used with PyTorch. The process is somewhat less complicated with 2D image data and simpler still with non-image data.

The proposed system then identifies the voxels of potential tumors in the lungs using PyTorch to implement a technique known as segmentation. This is roughly akin to producing a heatmap of areas that should be fed into the classifier in step. This ensures that the focus is on potential tumors inside the lungs and ignores huge swaths of uninteresting anatomy.

Then group interesting voxels into lumps: that is, candidate nodules (see figure 9.5 for more information on nodules). Here, the rough center of each hotspot on the heatmap is identified. Each nodule can be located by the index, row, and column of its center point. This is done to present a simple, constrained problem to the final classifier. Grouping voxels will not involve PyTorch directly, which is why this is pulled out into a separate step.

The next step classifies the candidate nodules as actual nodules or non-nodules using 3D convolution. The features that determine the nature of a tumor from a candidate structure are local to the tumor in question, so this approach should provide a good balance between limiting input data size and excluding relevant information.

Diagnose the patient using the combined per-nodule classifications. Similar to the nodule classifier in the previous step, an attempt is made to determine whether the nodule is benign or malignant based on imaging data alone. A simple maximum of the per-tumor malignancy predictions is taken, as only one tumor needs to be malignant for a patient to have cancer.

1.4 Advantages of the Proposed System

- Detecting lung cancer early has a huge impact on survival rate
- Early detection, screening, and diagnosis have been proven to significantly improve patient survival rates and quality of life.
- Significantly reduce the cost and complexity of cancer treatment.
- Three dimensions are better than two. 3D images mean that the x-rays produced come with more detail than their 2D counterparts.

Chapter 2

LITERATURE REVIEW

As per the discussions in paper [1], it aims to reproduce the state-of-the-art full-volume model for lung cancer risk prediction. This model is reported to exceed the performance of expert radiologists on this task. The model's architecture is an Inflated 3D ConvNet (I3D) which is based on 2D ConvNet inflation: filters and pooling kernels of very deep image classification ConvNets are expanded into 3D, making it possible to learn seamless spatio-temporal feature extractors from 3D image volumes while leveraging successful ImageNet architecture designs and even their parameters. This approach leverages a deep convolutional neural network (CNN) to automate this complex image analysis task. The two main components are the pre-processing module, which segments and centers the lungs and the full-volume model, a 3D CNN trained to take the entire lung volume and produce a cancer risk prediction score for that patient. National Lung Cancer Trial (NLST) dataset is used which is the largest publicly available chest LDCT dataset, with 25,000 participants. Area Under the (ROC) Curve (AUC) is used as the main evaluation metric. A small subset of NLST is used to train the model and a state-of-the-art AUC score of 0.892 is achieved which is a testament to the effectiveness of the I3D architecture.

This study [2] provides a lung cancer detection system based on CT-scan images. This detection system has 4 main stages, namely pre-processing of CT-Scan images to improve image quality, segmentation to identify and separate the desired cancer object from the background, feature extraction based on area, contrast, energy, entropy, and homogeneity. The segmentation method used in this research is thresholding which aims to divide the histogram of an image with different gray levels into two areas without having to enter a threshold value. The approach taken uses the discriminant analysis method by determining the variables that can differentiate between two or more groups. The segmentation phase uses the find contour method, using the contour Area function in OpenCV. The GLCM method is used to calculate the features from the cancer object. The classification stage is carried out using the Support Vector Machine (SVM) method. 35 CT scan images are used; there are 10 normal data, 20 malignant lung cancer data, and 5 benign lung cancer data. From the system trial, the accuracy level was 83.33%.

In this research study [3], an automated approach has been proposed where CT Scan gray-scale images were incorporated for cancer detection. The pre-processing consists of enhancement, filter operation, and segmentation. The post-processing consists of feature extraction and identification. The dataset contains 000 T1-weighted contrast-enhanced images from 3 patients. This research not only contract down the approaches which reduce the time but also gives the greater accuracy comparing to others. But it is still possible to increase the accuracy rate and hence there is still opportunity to work with gray-scale images. 3D CT scans will give better results compared to 2D. Thus, this study proposed approach was able to achieve an accuracy rate of 95% which is acceptable in the pathology Laboratory.

Using CNN as an extractor and classifier, this research [4] suggests a technique for classifying lung nodules in either benign or malignant. The system proposed in this paper is divided into three steps. The first step was the acquisition of CT images. The second step of segmentation was carried out by using expert marking. The third and the final step was diagnosis by the CNN model indignant or benign tissue. The author of this paper has made use of LIDC and IDRI database along with the 1018 test of CT included online. Among the 1081 images, 185 weren't used due to two reasons: incorrect marking and examinations display nodules of $\leq$ 3mm. The proposed CNN architecture obtained 97.2% accuracy, 95.6% sensitivity, and 96.1% specificity. Another important point to be noted from this paper is the application and analysis, without computing the morphology and texture.

This paper [5] uses a Deep Convolutional Neural Network (DCNN) and numerous pre-processing techniques to build up the exactness of the automated prediction of Lung nodules and their malignancy using CT scans. After training the model and minimizing the loss function using stochastic gradient descent (SGD), the final model was rendered to pinpoint the coordinates of malignant nodules, predict the probability of it being malignant and estimate its malignancy. The attempt of using the C3D architecture has increased the sensitivity of Malignant Lung Nodule Detection to 86%, which is 10% more than the previous effort at it. This progress was achieved through advanced pre-processing, additional training, and better prediction techniques. Dataset used: LIDC-IDRI dataset and resources obtained from the LUNA16. The sensitivity of Malignant Lung Nodule Detection obtained is 86%.

This paper [6] identifies the problem of discrimination between benign and malignant nodules on screening and non-screening scans. A machine learning-based lung cancer benign/malignant classification pipeline is proposed, which co-learns with 3D CT image volumes and clinical demographics. An attention-based CNN is proposed in this paper to train and validate the 3D CT image scan. The 2D soft attention gate (SAG) was extended to 3D in the proposed 3D network. A random forest (RF) classifier, called "Xgboost", has been used to learn from (1) clinical features only and, (2) both image and clinical features. The proposed Co-learning model achieved superior performance on both validation and testing data compared with the baseline methods (e.g., the Co-learning model improves the AUC from 0.687 (image-only) and 0.635 (clinical-only) to 0.787). The major limitation of the work is the limited number of training scans, which is relatively small for deep learning.

The data used in this paper [7] is the Cancer Imaging Archive database where images are stored in DICOM format. The image database contains CT images of patients with and without lung cancer. Image preprocessing is utilized to suppress unwanted distortions and enhance the image. Image smoothing using median filtering and image enhancement is done to improve the quality of the digital images. Image Segmentation is used to separate out the required region of interest. Here, preprocessed grayscale images are converted to binary images. Morphological opening operation is performed to the binary image with disk structuring element for removal of unwanted components from the image. Lung masks are obtained by filling the holes gaps present in the lungs. Feature extraction is done on the segmented region of interest. These features serve as input for the classification of CT scan images. The size and shape of the tumor present in the lungs are estimated by extracting three geometrical features: Area, Perimeter and Eccentricity. Classification involves labelling the CT scan images as normal and an abnormal using SVM algorithm.

In this paper [8], four two-pathway Convolutional Neural Networks (CNN) are proposed, including a basic 3D CNN, a novel multi-output network, a 3D DenseNet, and an augmented 3D DenseNet with multi-outputs. These four networks are evaluated on the public LIDC-IDRI dataset and outperform most existing methods. In particular, the 3Dmulti-output DenseNet (MoDenseNet) achieves the state-of-the-art classification accuracy on the task of end-to-end lung nodule diagnosis. In addition, the networks pretrained on the LIDC-IDRI dataset can be further extended to handle smaller datasets

using transfer learning. There are 147 CT scans (37% benign and 63% malignant collected for this lung cancer diagnosis problem. The location for each pulmonary nodule is specified by the radiologist and each nodule was determined whether it is benign or malignant by taking a biopsy of the nodule. The basic 3D CNN can is improved by providing intermediate outputs and/or adding connections between layers, which shorten the distance from input to output and in turn achieve better optimization results. The multi-output DenseNet obtains the highest accuracy of 90.40% and area under the curves (AUC) of 0.9548.

This paper [9] proposes a Recurrent Convolutional Neural Network (RCNN) based on U-Net as well as a Recurrent Residual Convolutional Neural Network (RRCNN) based on U-Net models, which are named RU-Net and R2U-Net respectively. This research demonstrates two modified and improved segmentation models, one using recurrent convolution networks, and another using recurrent residual convolutional network. The experiments are conducted on three different modalities of medical imaging including retina blood vessel segmentation, skin cancer segmentation, and lung segmentation. Performance evaluation of the proposed models is conducted for the patch-based method for retina blood vessel segmentation tasks and the end-to-end image-based approach for skin lesion and lung segmentation tasks. Comparison against recently proposed state-of-the-art methods that shows superior performance against equivalent models with the same number of network parameters.

This paper [10] proposes a 3D multipath VGG-like network, which is evaluated on 3D cubes, extracted from Lung Image Database Consortium and Image Database Resource Initiative (LIDC-IDRI), LUng Nodule Analysis 2016 (LUNA16), and Kaggle Data Science Bowl 2017 datasets. 3D multipath VGG-like network is proposed with 2 classifications. One classification is of lung nodules and non-nodules and the other is of benign nodules and malignant nodules. This study uses VGG-16-like network with convolutional and pooling layers called VGGNet, as it is lightweight and trains faster. The predictions from the VGGNet architectures are further combined with the segmentation model. Using U-Net architecture, segmentation masses are generated for lung CT scan images and lung nodules are segmented. This approach gives an accuracy of 95.66% and loss of 0.09 and dice coefficient of 90% and for predicting log loss is 38%.

The paper [11] presents a fully automated lung computed tomography (CT) cancer diagnosis system, DeepLung. DeepLung consists of two components, nodule detection (identifying the locations of candidate nodules) and classification (classifying candidate nodules into benign or malignant). A 3D Faster Regions with Convolutional Neural Net (R-CNN) is designed for nodule detection with 3D dual path blocks and a U-net-like encoder-decoder structure to effectively learn nodule features. For nodule classification, gradient boosting machine (GBM) with 3D dual path network features is proposed. The nodule classification subnetwork was validated on a public dataset from LIDC-IDRI, on which it achieved better performance than state-of-the-art approaches and surpassed the performance of experienced doctors based on image modality. Within the DeepLung system, candidate nodules are detected first by the nodule detection subnetwork, and nodule diagnosis is conducted by the classification subnetwork. Finally, gradient boosting machine with combined features are trained to classify candidate nodules into benign or malignant with an accuracy of 81.41%.

In this paper [12], a lung cancer detection algorithm is proposed using mathematical morphological operations for segmentation of the lung region of interest, from which Haralick features are extracted and used for classification of cancer by artificial neural networks. The methodology used is Data Loading, Pre-processing, Segmentation, Feature Extraction, and classification. The dataset used are images that were collected from the V.M.Salgaocar hospital, SMRC, and the Manipal hospital both situated in Goa. Cropping and Median filters are applied to the images, to get rid of the noise. The image is resized into three different resolutions followed by applying Haar wavelet transforms to these images. The GLCM (Gray Level Co-occurrence Matrix) function distinguishes the texture of an image, after which second-order statistical features or the Haralick features are extracted. The training accuracy was 96% and testing accuracy was 92%. The sensitivity was 88.7%. In this paper, only 128 training images are used which is relatively small for a deep learning problem especially in the biomedical domain.

The method proposed in this paper [13] is a 3D deep neural network to solve the automatic diagnosis of lung cancer from CT The model consists of two modules. The first one is a 3D region proposal network for nodule detection, which outputs all suspicious nodules for a subject. The second one selects the top five nodules based on the detection confidence, evaluates their cancer probabilities, and combines them with a leaky noisy-or gate to obtain the probability of lung cancer for the subject. The data is taken from two

datasets, LUNA and DSB. The nodules smaller than 6 mm from LUNA annotations are removed and nodules in DSB are manually labelled. The over- fitting caused by the shortage of training data is alleviated by training the two modules alternately and using 3D Data augmentation to increase the training data significantly. A 3D CNN is designed to detect suspicious nodules. It is a 3D version of the RPN using a modified U-net as the backbone model which enabled the network to capture multi-scale information, which is essential because the size of nodules has large variations. Accuracies achieved were 73.73% and 69.76% on the training and test sets.

In this paper [14], a network and training strategy is presented that relies on the strong use of data augmentation to use the available annotated samples more efficiently. The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. The network proved to give a very fast segmentation model. One important modification in the architecture is that the up-sampling part includes a large number of feature channels, which allows the network to propagate context information to higher resolution layers. The resulting network is applicable to various biomedical segmentation problems.

The paper [15] proposes MEMCAP, a deep neural network architecture rained with meta-learning to perform domain adaptation in lung nodule classification from CT scans. Datasets used are LUNA-16 Lung Nodule Dataset, Collected Lung Nodule Dataset and Collected Incidental Lung Nodule Dataset. It consists of a CapsNet feature network that extracts invariant low and high-level semantic structures across domains from the high-dimensional input volume. The output is then fed to the task network: a memory augmented recurrent network which learns to quickly store and retrieve domain-specific information from its external memory bank using a small number of labelled samples. MEMCAP is thus able to leverage the available labelled target examples to store and exploit the underlying intrinsic information in the target domain. The experimental results have demonstrated that when the data distribution changes, the proposed classifier adapts almost perfectly in the lung nodule classification task while popular deep networks' performance decrease with large domain shifts. MEMCAP outperforms all other methods and achieves 84.7% and 89.1% accuracy.

In this paper [16], the researchers have initiated to develop depth networks like convolutional neural network (CNN) and reinforcement learning model (RNN) as unsupervised lung tumor detectors, predictors, and classification modules. The latest lung nodule diagnosis, localization, and classifiers are done using the standard datasets LIDCIDRI, LUNA 16, and Super Bowl Dataset 2016 are familiar with supervised learning algorithms such as SVM, KNN, and CNN. Thus in order to lower the lung tumor death rate, this paper focused on constructing a cloud-based lung tumor detector that includes a segmentation module, detection module, and stage classifiers. The suggested Cloud-LTDSC module used unsupervised learning neural networks as a predictor and stage classifier to more precisely identify the tumor. The segmented images are feed-forwarded into the MCNN model in order to identify its classes and severity level for each patient. The e-record has been generated with patient name, severity level, and tumor details which are transferred through cloud to doctors for virtual monitoring and E-diagnosis automatically. The performance metrics are estimated in terms of accuracy, sensitivity, and specificity and compared with the existing techniques thus achieving an accuracy rate of 97%.

Chapter 3

ANALYSIS

3.1 Problem Identification

Lung cancer is listed as one of the most outrageous disorders in developing nations. Lung cancers, on average, double in size in four months to five months. Detecting lung cancer early has a huge impact on survival rate, but is difficult to do manually, especially in any comprehensive, whole-population sense. Currently, the work of reviewing the data must be performed by highly-trained specialists. This requires painstaking attention to detail, and it is dominated by cases where no cancer exists. Searching this way results in the potential for missed warning signs, particularly in the early stages when the hints are more subtle. The human brain just isn't built well for that kind of monotonous work.

Fig 3.1 Lung Cancer Statistics

Source: American Source Society, 2017

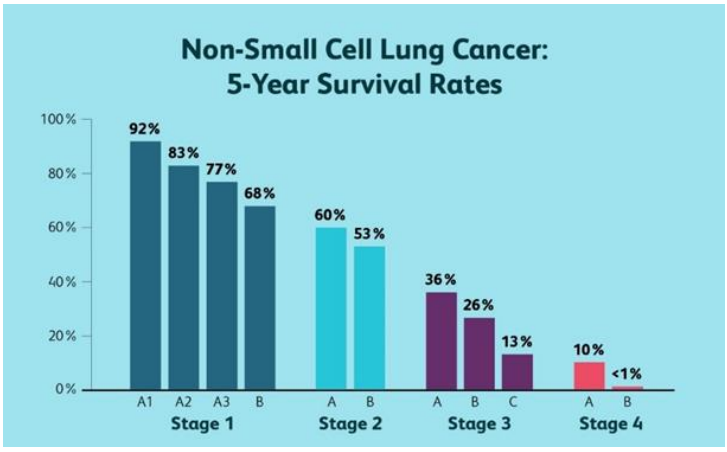


Fig 3.2 Lung cancer survival rates

Source: American Cancer Society, 2017

Lung cancer detection has earlier been overworked using image processing approaches. Computer-aided diagnosis (CAD) is suitable for this task because computer vision models can quickly scan everywhere with equal quality and they are not affected by fatigue and emotions.

In the direction of preventing the Lung cancer disease and for analyzing the early stage of this disease required powerful technology to assist the doctors is most desirable, in particular Image Processing, Machine Learning, and Artificial Intelligence technique can process the medical field data with the aid of engineering solution for the purpose of detection and diagnosing the lung disease.

This project identifies the limitations of hand-engineered techniques for lung cancer detection and is an attempt at automating the detection process using Deep Learning. The end-to-end approach for the detection and classification of objects is very successful in general vision tasks. The project architecture used in the project has the benefit of working well with a more modest amount of data. Here, two neural network models are being employed in order to perform image segmentation and nodule analysis and classification. There are five main steps involved from examining a whole-chest CT scan to giving the patient a lung cancer diagnosis.

3.2 Objectives

- This project aims to develop a lung cancer detection system based on 3D CT-scan images.
- The objective of this project is to design an automatic detection and classification system that would identify cancerous tumors accurately which otherwise would be tedious compared to traditional hand-engineered methods.
- The project aims to build a 3D Convolutional neural network model that works best for classification of the lung nodule as benign or malignant.
- It also aims to build a segmentation model with Convolution Neural Networks using U-Net as a backbone so as to increase the ease of analysis of these images by the classification model. It also performs grouping as a separate task to identify and group interesting data from the segmentation output to be fed into the classifier.

- To tackle this cutting-edge project effectively, the project will make use of PyTorch. PyTorch is a library for Python programs that facilitates building deep learning projects. It emphasizes flexibility and allows deep learning models to be expressed in idiomatic Python.
- Explore pre-processing, transformation techniques to understand nonstandard data without prebuilt libraries and create training samples suitable to plug into a model such as a classification model.

3.3 Methodology

A CT scan is more likely to show lung tumors than routine chest x-rays. It can also show the size, shape, and position of any lung tumors and can help find enlarged lymph nodes that might contain cancer that has spread.

The CT data files will be loaded to produce a CT instance that contains the full 3D scan, combine that with a module that performs segmentation (flagging voxels of interest), and then group the interesting voxels into small lumps in the search for candidate nodules. The nodule locations are combined back with the CT voxel data to produce nodule candidates, which can then be examined by the nodule classification model to determine whether they are actually nodules in the first place and, eventually, whether they're malignant.

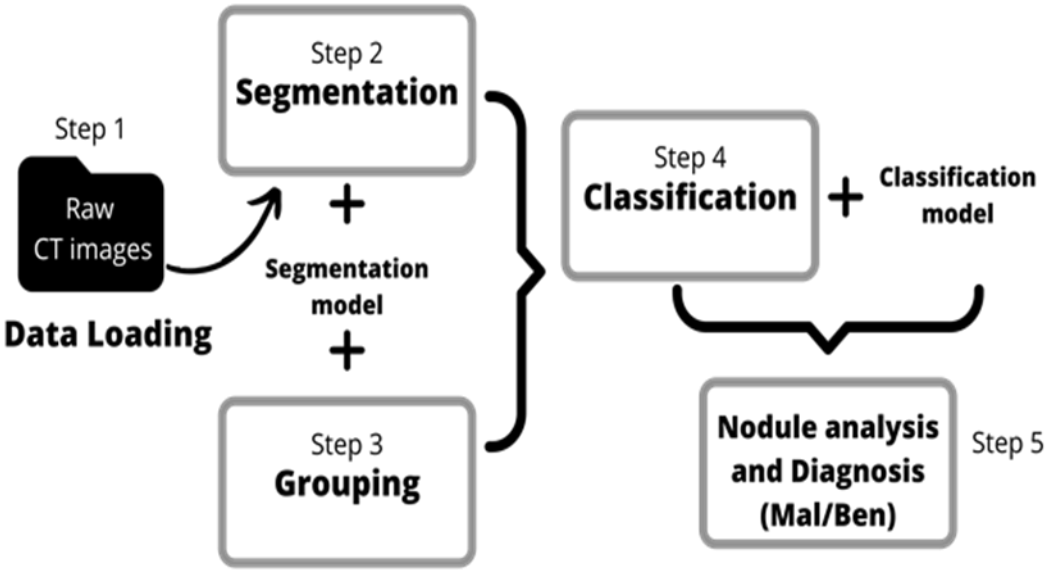


Fig 3.3 Methodology

3.3.1 Data Loading

The raw CT scan data has to be loaded into a form that can be used with PyTorch. The CT data comes in two files: a .mhd file containing metadata header information, and a .raw file containing the raw bytes that make up the 3D array. The native file format for CT scans is DICOM. Each specific CT scan is identified using the series instance UID assigned when the CT scan was created, according to the DICOM nomenclature. The annotation data provided by LUNA will also be loaded, which will give a list of nodule coordinates, each with a malignancy flag, along with the series UID of the relevant CT scan. All the data transforms required to make a sample tuple are performed. These sample tuples will be used as input to the model training routine.

3.3.2 Segmentation

This step aims to identify the voxels of potential tumors in the lungs using PyTorch to implement a technique known as segmentation. Image segmentation is a computer vision task in which specific regions of an image are labelled according to what's being shown. More specifically, the goal of semantic image segmentation is to label each pixel of an image with a corresponding class of what is being represented. Because it is predicting for every pixel in the image, this task is commonly referred to as dense prediction.

A segmentation model has repeated layers of convolution and down sampling where the model starts by consuming raw pixels to produce specific, detailed detectors for things like texture and color, and then builds up higher-level conceptual feature detectors for parts like eyes. Since segmentation needs to produce an image-like output, a pre-existing U-Net is integrated into the segmentation model which uses a Fully Convolutional Network Model for the task. With the U-Net model, prediction can be collapsed into a segmentation map by taking the argmax of each depth-wise pixel vector. The target can be easily inspected by overlaying it onto the observation.

3.3.3 Grouping

The segmentation model will predict if a pixel is part of a nodule. This will be done per 2D slice, and every 2D result will be stacked to form a 3D array of voxels containing nodule candidate predictions. This step involves grouping the voxels into lumps: i.e., nodule candidates by applying a threshold to the predictions, and then grouping connected regions of flagged voxels. This process will find the rough center of each hotspot on the heat map.

Each identified nodule candidate will be used to construct a sample tuple for classification. Each nodule can be located by the index, row, and column of its center point. This is done to present a simple, constrained problem to the final classifier. Grouping voxels will not involve PyTorch directly, which is why it is pulled out into a separate step.

3.3.4 Classification

The nodule candidates just produced by grouping will be passed to the candidate classification model after which malignancy detection can be performed on the candidates flagged as nodules. Each nodule candidate from segmentation and grouping will be classified as either nodule or non-nodule. Doing so will permit screening out the many normal anatomical structures flagged by the segmentation process. This process is done using 3D Convolution Neural Network.

The features that determine the nature of a tumor from a candidate structure are local to the tumor in question, so this approach should provide a good balance between limiting input data size and excluding relevant information. Making scope-limiting decisions like this can keep each individual task constrained, which can help limit the amount of things to examine when troubleshooting.

3.3.5 End-to-end detection

Finally, all the above processes are put together to get to the finish line, combining the components into an end-to-end solution that can look at a CT and answer the question “Are there malignant nodules present in the lungs?” The last step is to diagnose the patient using the combined per-nodule classifications. Similar to the nodule classifier in the previous step, the model will attempt to determine whether the nodule is benign or malignant based on imaging data alone. A simple maximum of the per-tumor malignancy predictions is taken, as only one tumor needs to be malignant for a patient to have cancer.

3.4 System Requirements

3.4.1 Hardware requirements

Processor: Intel i7 10th generation or higher/ AMD Ryzen

GPU: CUDA-capable GPU (NVIDIA GTX 1070 or better)

RAM: 8 GB minimum

Free disk space: 200 GB (120 GB uncompressed data + 80GB training)

3.4.2 Software requirements

OS: Windows 10

Language: Python

Front-end: React

Back-end: Flask

Libraries: PyTorch, SimpleITK, Scipy etc

IDE: Jupyter Notebook, VS code

3.5 Functional Requirements

A Functional Requirement defines a function of a system or its component, where a function is described as a specification of behaviour between outputs and inputs. These are usually set by the end users.

1. The user/expert must be able to perform analysis on the desired CT scan images with ease and without any delay in obtaining the results.
2. The user/expert must be able to download the final results of the data for future use.
3. The user must be allowed to upload only .mhd and .raw files of the pulmonary CT scans.

3.6 Non-Functional Requirements

1. Performance: The PCs used must at least be INTEL CORE i7 machines so that they can give optimum performance of the product.
2. Reliability: The system should have little or no downtime.
3. Ease of Use: The user views should be easy to use and intuitive.

Chapter 4

SYSTEM DESIGN

4.1 System Architecture

4.1.1 Classification Model: 3D Convolutional Neural Network

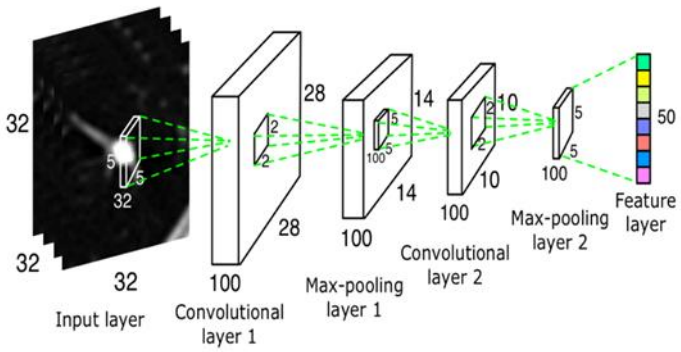


Fig 4.1 CNN Architecture

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

4.1.2 Convolution Layer – the Kernel

The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter. The Kernel shifts a certain number of times based on Stride Length, every time performing a matrix multiplication operation between the kernel and a portion of the image over which it is hovering.

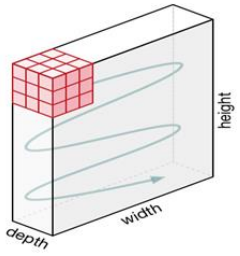


Fig 4.2 Movement of Kernel

4.1.3 Pooling Layer

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. Max Pooling also performs as a Noise Suppressant. Moving on, the final output can be fed to a regular Neural Network for classification purposes.

4.1.4 Classification – Fully Connected Layer (FC Layer)

The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique for the binary classification task. Softmax is an activation function in a neural network model. It is used to normalize the outputs, converting them from weighted sum values into probabilities that sum to one. Each value in the output of the softmax function is interpreted as the probability of membership for each class.

4.1.6 U-Net Architecture for Segmentation

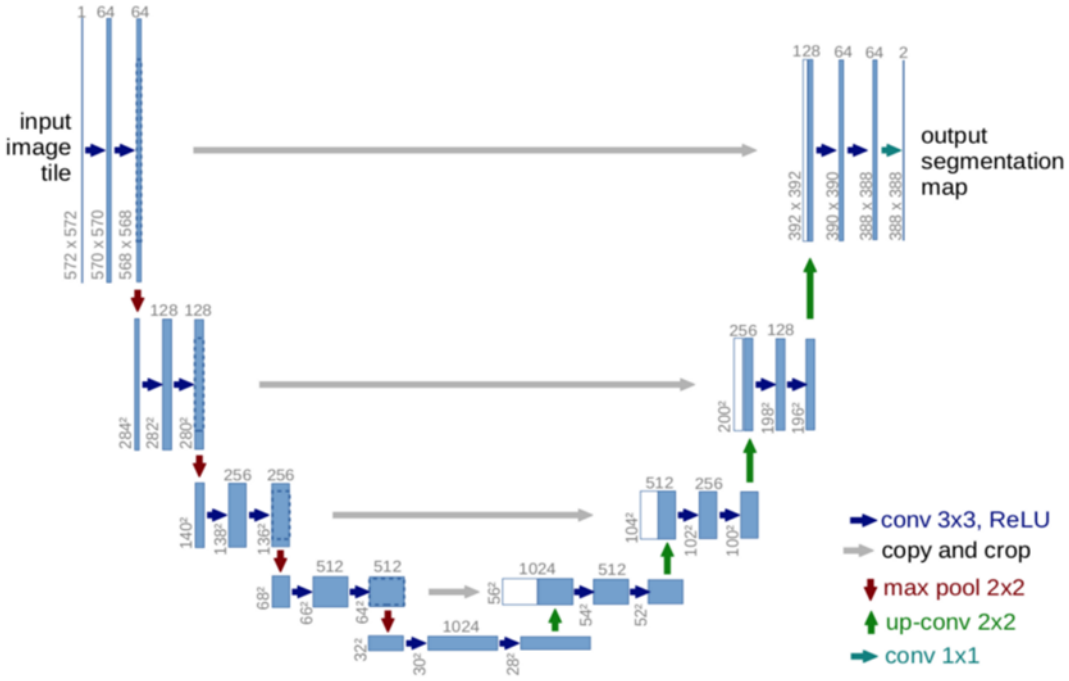


Fig 4.3 U-Net Architecture

U-Net is an architecture for semantic segmentation. The reason it is able to localise and distinguish borders is by doing classification on every pixel, so the input and output share the same size. It consists of a contracting path and an expansive path. The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step, the number of feature channels are doubled.

Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 convolution ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU.

Each process constitutes two convolutional layers. The red arrow pointing down is the max pooling process which reduces size of image. The data flows from top left to bottom center through a series of convolutions and downscaling. Transposed convolution is an upsampling technique that expands the size of images. It does some padding-so as to keep the image size same as input-on the original image followed by a convolution operation. In U-Net, skip connections short-circuit inputs along the downsampling path into the corresponding layers in the upsampling path. This means those final detail layers are operating with the best of both worlds. These skip connections from earlier layers in the network (prior to a downsampling operation) should provide the necessary detail in order to reconstruct accurate shapes for segmentation boundaries. More fine-grain detail can be obtained with the addition of these skip connections.

4.2 Detailed Design

In detailed system design, every system needs to be broken down to ascertain all activities required and their respective inputs and outputs. In some of the cases, sub systems are broadly defined in the conceptual design phase, but at this stage sub systems are specifically defined to work out every detail concerning the sub-system. Decomposition of the system to operational activities in general is carried out as follows.

4.2.1 High Level Design

The proposed approach to detecting cancerous nodules will have five rough steps: data loading, segmentation, grouping, classification, and nodule analysis and diagnosis.

The first step is to convert the 3D CT scans into a format that the PyTorch library can understand. Following the conversion of the data into a format that the system can comprehend voxels of potential tumors in the lung CT data are detected using a segmentation method that masks these tumors for simple detection. The proposed system utilizes semantic segmentation which is per-pixel labelling to flag or mask the interesting voxels. Then, in order to find candidate nodules, aggregate the interested voxels into tiny lumps. The nodule locations are paired with CT voxel data to create nodule candidates, which may then be reviewed by the system's next phase, the nodule classification model, to see if they're nodules in the first place and, eventually, if they're cancerous.

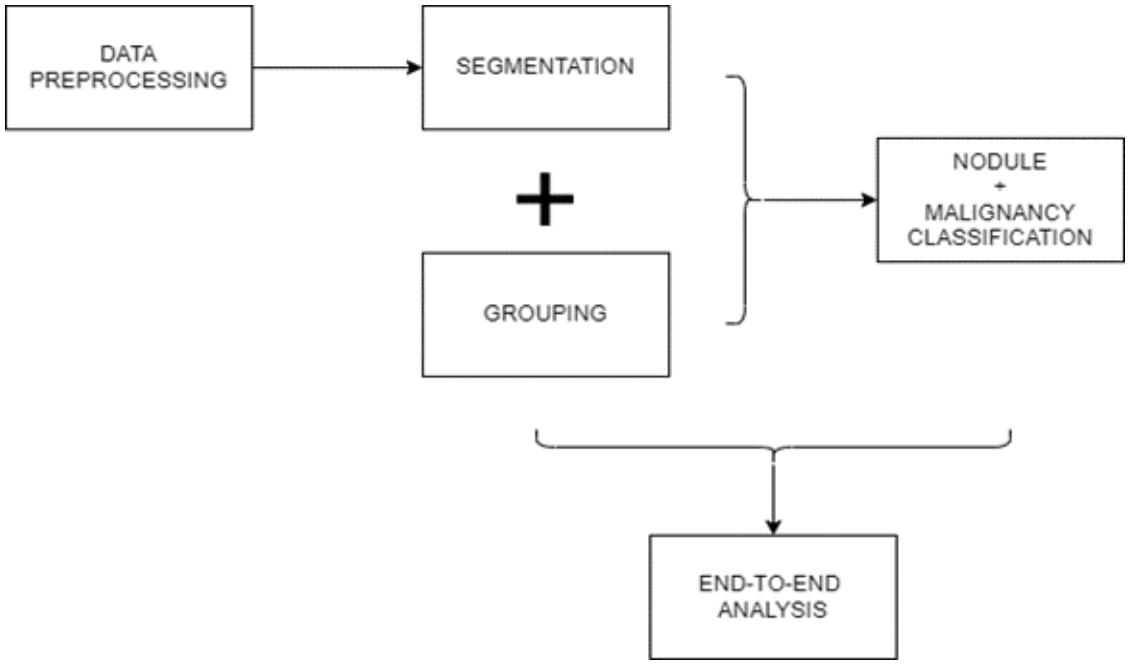


Fig 4.4 High level design diagram

4.2.2 Low Level Design

4.2.2.1 Data Loading

The first and the foremost step in the process is the need to be able to take CT data from a pile of bits on disk and turn it into a Python object from which 3D nodule density data is the extracted. As depicted in the figure given below, the bits represented as the .mhd and .raw files are converted to CT objects. The format of the data files is treated as a black box and a library called SimpleITK is used to load them into 3D NumPy arrays. The dataset being used has 10 subsets that have about 90 CT scans each (888 in total), with every CT scan represented as two files: one with a .mhd extension and one with a .raw extension.

Specific CT scans are identified using the series instance UID (series_uid) assigned when the CT scan was created. The coordinates are then to be transformed from the millimeter-based coordinate system (X, Y, Z) they're expressed in, to the voxel-address-based coordinate system (I, R, C) used to take array slices from the CT scan data.

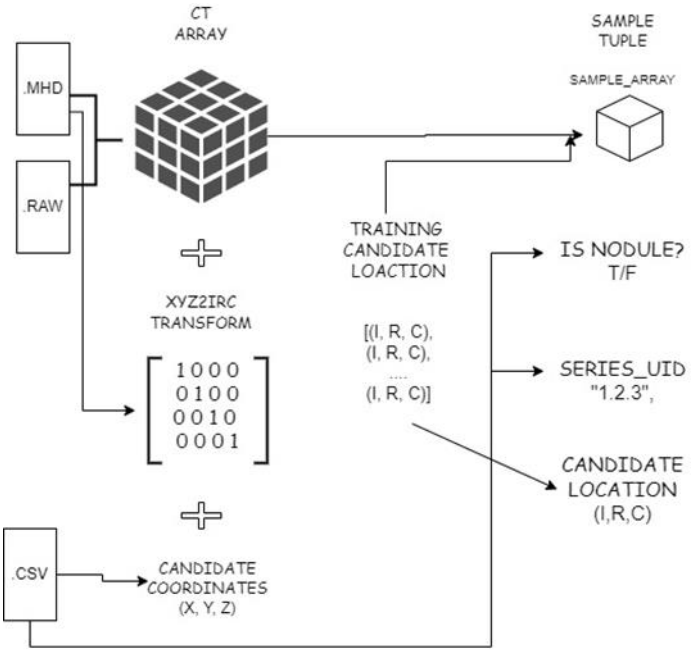


Fig 4.5 Data Loading diagram

A match between the (X, Y, Z) coordinates and the coordinates given in the .csv file is performed which returns a cubic chunk of CT which is labelled as sample array in the image shown below, as well as the center of the candidate converted to array coordinates. A four-item sample tuple is finally returned which contains the following items: the sample array, a nodule flag represented as Boolean values, series_uid of each candidate and the candidate location in the (I, R, C) format.

4.2.2.2 Segmentation

The segmentation process uses a pre-existing U-Net implementation. This U-Net model is adapted to work well with the LUNA data set. The total depth and the number of filters is reduced to reduce the memory consumption. Due to the fact that pixel spacing in the z direction is much larger than in plane the nodule is less likely to be present across many slices. This makes a fully 3D approach less attractive for this projects purpose.

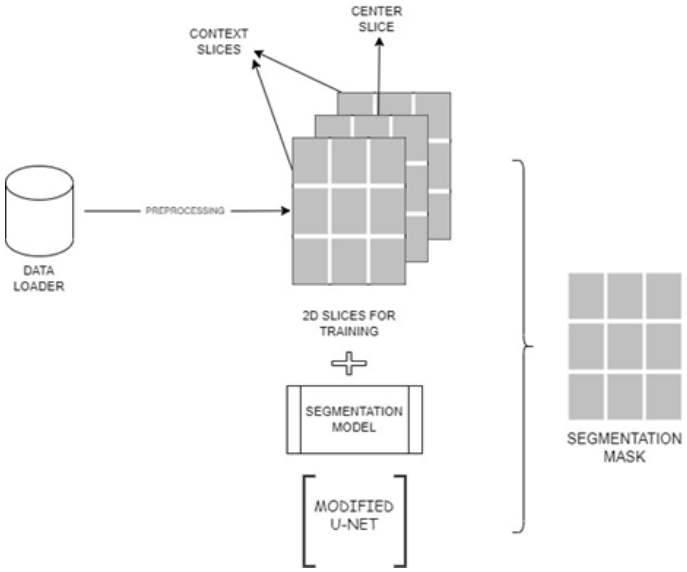


Fig 4.6 Segmentation Diagram

In segmentation, instead of training the model with 3D data, each slice is treated as a 2D segmentation problem. To work around the issue of context in third dimension several context slices or neighbouring slices are provided along with the main slice. The data loader will return two different types of data sets for training and validation. The training will be performed on 2D slices, and the validation is performed either on the full CT slice or on the crops.

During segmentation training the augmentation is done during the forward propagation on the GPU to speed up the process. These 2D slices combined with the segmentation model provide the output segmentation mask which is essentially a heat map of the CT scan which shows the location of nodules flagged as positive.

A simple connected components algorithm is used for grouping the suspected nodule voxels into chunks to feed into classification. During grouping the centre coordinates of the nodules are also calculated and returned as output in the list of tuples. The centre coordinates are used to obtain 3D crops of the nodules for classification. The grouping process does not remove anything explicitly, it reduces the number of items considered since voxels are consolidated into nodule candidates.

4.2.2.3 Classification

a) Nodule classification

The application's functionality is implemented via a class so that the application can be instantiated and passed around if the need be. The first step, is to initialize the model and optimizer; and the second is to initialize the Dataset and DataLoader instances.

LunaDataset will define the randomized set of samples that will make up the training epoch, and the DataLoader instance will perform the work of loading the data out of the dataset and providing it to the application. The PyTorch Data-Loader class will handle all of the collation work. During training, the best state models are saved. These best models are loaded to be used for predictions.

The data from the data loader is received in batches and is provided as input to the forward propagation after nodule and non-nodule classifier. The output of the classification model contains a list of candidates which are flagged as nodules by the model. The output provides probabilities associated with each candidate specifying how likely they are to be positive nodules. This data is done forwarded to the malignancy classifier.

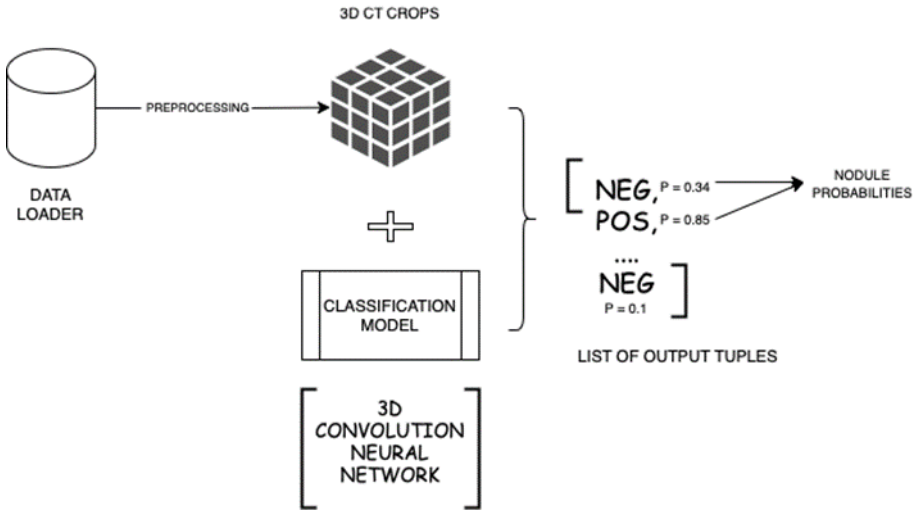


Fig 4.7 Nodule Classification diagram

b) Malignancy Classifier

The malignancy classifier uses a different set of data than the nodule and non-nodule classifier. This classifier is only trained on the positive nodules. The malignancy classifier data set contains malignant and benign nodule data. The data set is augmented using different augmentation techniques to bring about a balanced ratio and is returned in batches to the malignancy classifier. This data like the classification data set contains 3D crops of the nodules from the CT scan. The malignancy classifier model is trained using fine tuning on the nodule non-nodule classification model. Fine tuning uses the existing parameters from the nodule non nodule classification model, and only trains the head part of the convolutional neural network. This model is used along with the data set from the data loader to make predictions.

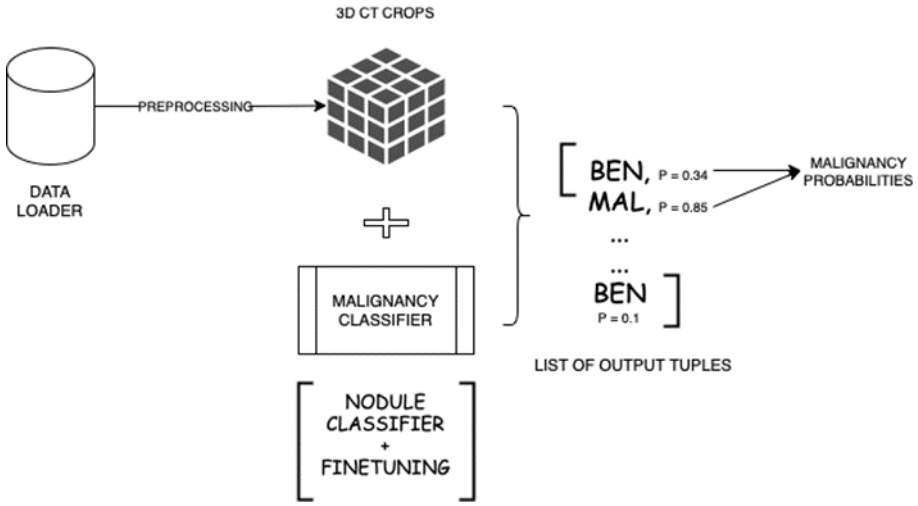


Fig 4.8 Malignancy Classification diagram

4.3 Dataflow Diagram

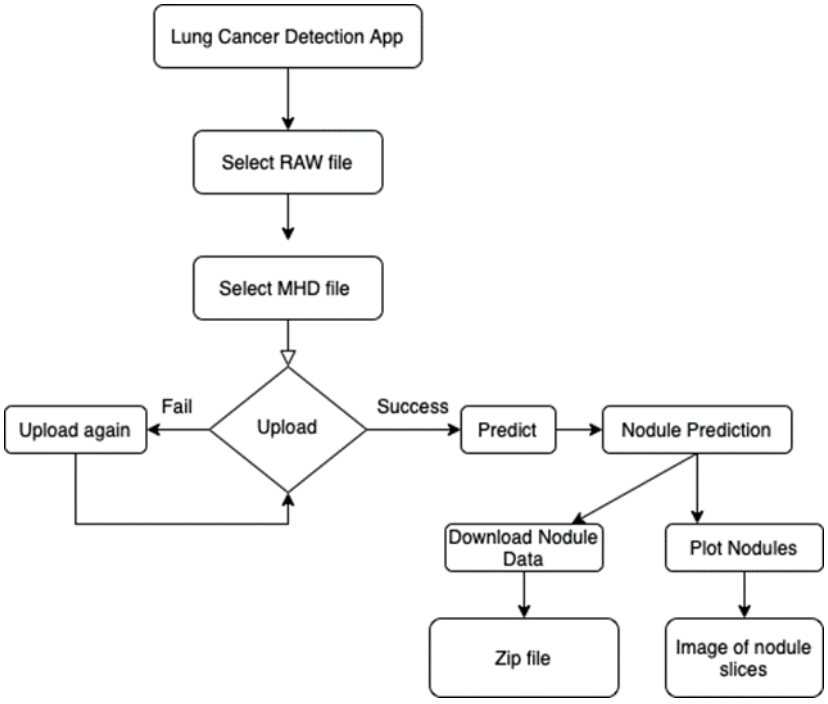


Fig 4.9 Dataflow diagram

4.4 Use-Case Diagram

Use cases are a way of describing interactions between users and a system using a graphical model and structured text. In their simplest form, a use case identifies the actors involved in an interaction and names the type of interaction. Use cases are documented using a high-level use case diagram. The set of use cases represents all of the possible interactions that will be described in the system requirements.

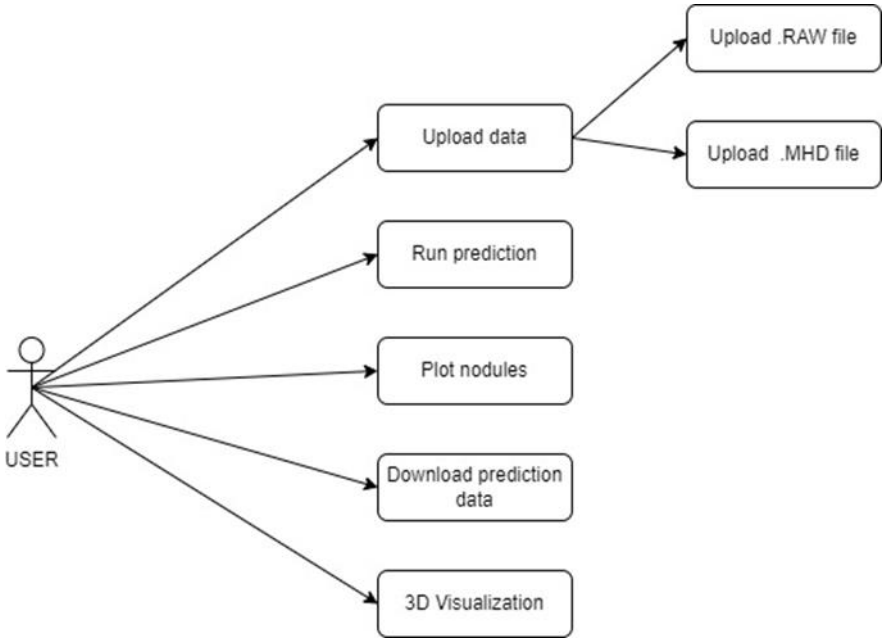


Fig 4.10 Use case diagram

Table 4.1 Use case table

LUNA - Lung Cancer Detection	
Actors	Radiologists
Description	The CT scan data can be uploaded to the system by a radiologist. The technology analyses the image data and determines the position of any nodules that may exist. If nodules are discovered, they are grouped and entered into a classification model that forecasts the likelihood of cancer. The radiologist will also receive additional information, such as the coordinates of the possible nodule and picture plots depicting the nodule's position. The radiologist may download the cropped CT scan data and use third-party software to visualise it.
Data	3D CT Image Data (.mhd and .raw files)
Stimulus	User command issued by the radiologist
Response	Identifying whether a nodule is present, as well as the likelihood of the nodule's malignancy
Comments	The radiologist must have the necessary files to upload, which include the metadata and the CT scan's RAW image.

4.5 Sequence Diagram

Sequence diagrams are primarily used to model the interactions between the actors and the objects in a system and the interactions between the objects themselves.

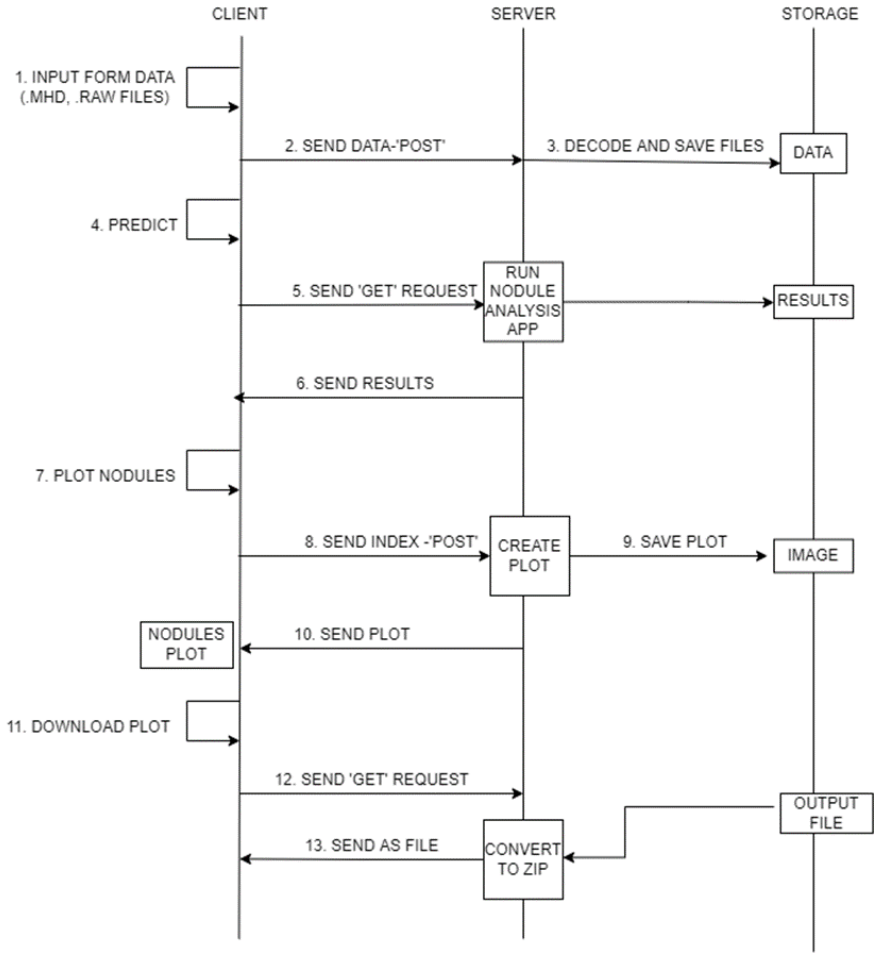


Fig 4.11 Sequence diagram

The form data, which consists of the .mhd and .raw files, is entered by the client. The API is triggered when the form data is submitted, and the files are uploaded to the server. The files are decoded and saved in the data directory on the server. The data is ready for inference after the upload process is completed successfully. The client then instructs the API to apply the trained models to the newly submitted data. The models

run and produce results, which are presented to the client in a structured manner. These results can be used by the client for analysis. The client may now plot images for various nodules and download nodule data from the server in the form of an archive. The API facilitates these tasks once again. A third-party tool can be used to extract and view the archived nodule data.

Chapter 5

IMPLEMENTATION

5.1 Overview of System Implementation

The proposed system first deals with data loading and pre-processing and then jumps to classification before implementing segmentation and grouping, as classification requires an approach using multiple convolutional and pooling layers to aggregate spatial information before feeding it into a linear classifier.

When compared to having to look at the full image at once, being able to zoom in and focus on the detail of a specific place will have a big impact on overall productivity while training the model. The segmentation model is required to consume the entire image, but modules are also structured so that the classification model gets a zoomed-in view of the areas of interest.

Later grouping will produce and classification will consume data containing sequential transverse slices of a tumor. This image is a close-up view of a (potentially malignant, or at least indeterminate) tumor, and it is what will be used to train the nodule classification model to identify, and the malignancy classification model to classify as either benign or malignant.

5.2 Algorithm

5.2.1 Image Classification

Image classification can be defined as the task of categorizing images into one or multiple predefined classes. Although the task of categorizing an image is instinctive and habitual to humans, it is much more challenging for an automated system to recognize and classify images.

A CNN is a framework developed using machine learning concepts. CNNs are able to learn and train from data on their own without the need for human intervention. There is only some pre-processing needed when using CNNs. They develop and adapt their own image filters, which have to be carefully coded for most algorithms and models. CNN frameworks

have a set of layers that perform particular functions to enable the CNN to perform these functions. This project utilizes CNN for two different types of classification - nodule classification and malignancy classification.

Nodule classification and malignancy classification-
The nodule candidates are passed to the candidate classification step, and then malignancy detection is performed on the candidates flagged as nodules.

a) Nodule classification—Each nodule candidate from segmentation and grouping will be classified as either nodule or non-nodule.
b) Fine-tuning the malignancy classifier- a model specifically for classifying benign and malignant nodules is defined. The training process is carried out by fine-tuning: a process that cuts out some of the weights of an existing model and replaces them with fresh values that is then adapted to the new task.

5.2.2 Segmentation

It is vital that the classifier knows where exactly to look. To do this, the raw CT scans are taken and everything that might be a nodule is identified. To find these possible nodules, voxels that look like they might be part of a nodule must be flagged, a process known as segmentation. The code is structurally very similar to what was developed for classification, but differs in detail:

1. A pre-existing U-Net is integrated into the segmentation model which outputs an entire image.
2. The dataset being utilised must be changed so as to not only deliver bits of the CT but also provide masks for the nodules. The classification dataset consisted of 3D crops around nodule candidates, but it is required to collect both full CT slices and 2D crops for segmentation training and validation.
3. A new loss function is utilized so as to adapt and optimize the training process.

For this project, semantic segmentation is used, which is the act of classifying individual pixels in an image using labels. If done properly, this will result in distinct chunks or regions. This takes the form of a label mask or heat map that identifies areas of interest. A simple binary label is obtained: true values will correspond to nodule candidates, and false values mean uninteresting healthy tissue.

5.3 Pseudocode

5.3.1 Prediction

```
def main(self, image_data_path=os.path.join(ROOT_DIR, "data")):

    file = glob.glob(os.path.join(image_data_path, "*.mhd"))[0]
    ct = Ct(file)  # done
    mask_a = self.segmentCt(ct)  # done

    candidates_list = self.groupSegmentationOutput(ct, mask_a)
    classifications_list = self.classifyCandidates(ct, candidates_list)

    mal_list = []

    for tup in classifications_list:
        prob, prob_mal, center_xyz, center_irc = tup
        if prob > 0.5:
            mal_tup = Mal_Tup(prob, prob_mal, center_xyz, center_irc)
            mal_list.append(mal_tup)

    # visualize(ct, mal_list)
    result = convertOutput(mal_list)
    return ct, result

    # return ct, mal_list
```

All of the components are executed in a sequence to provide the predicted results. This provides an end-to-end solution that can look at a CT and answer the question “Are there malignant nodules present in the lungs?” In each step, the respective models are loaded with trained parameters.

Ct: Get the Ct array for prediction from the input files.
IRC: CT segmentation and grouping is done get nodule candidate samples to classify.
Determine the nodules: Nodule classification is performed on the candidate to determine whether it should be fed into the malignancy classifier.
Determine malignancy: Malignancy classification is performed on the nodules that pass through the nodule classifier to determine whether the patient has cancer.

5.3.2 Ct Class

```
class Ct:
    def __init__(self, image_data):

        ct_mhd = sitk.ReadImage(image_data)
        ct_arr = np.array(sitk.GetArrayFromImage(ct_mhd), dtype=np.float32)

        ct_arr.clip(-1000, 1000, ct_arr)
        self.hunits_arr = ct_arr

        self.origin_xyz = XyzTuple(*ct_mhd.GetOrigin())
        self.voxSize_xyz = XyzTuple(*ct_mhd.GetSpacing())
        self.direction_arr = np.array(ct_mhd.GetDirection()).reshape(3, 3)
```

The Ct class is implemented, that loads data from disk and provides access to cropped regions around points of interest. The data from .mhd and .raw files is read as an array using the SimpleITK library. The data being split between multiple files is hidden behind the sitk Routines. At this point, the extracted ct_arr is a three-dimensional array. All three dimensions are spatial, and the single intensity channel is implicit. In a PyTorch tensor, necessary for training, the channel information is represented as a fourth dimension with size 1.

The value at each voxel is stored in a 3D array as HU values in hunits_arr. Some CT scanners use HU values that correspond to negative densities to indicate that those voxels are outside of the CT scanner's field of view. For the purpose of this project, everything outside of the patient should be air, so that field-of-view information is discarded by setting a lower bound of the values to –1,000 HU. Similarly, the exact densities of bones, metal implants, and so on are not relevant to this use case, so density is capped at roughly 2 g/cc (1,000 HU).

The voxel size, direction matrix, and origin coordinates are also extracted from the mhd file. This metadata permits the construction of the transformation from patient coordinate system (X, Y, Z) to index, row and column (I, R, C) coordinates.

5.3.3 Segmentation

5.3.3.1 Dataset

```
class Luna2dSegmentationDataset(Dataset):
    def __init__(
        self,
        ct,
        contextSlices_count=3,
    ):

        self.contextSlices_count = contextSlices_count
        self.sample_list = []

        index_count = int(ct.hunits_arr.shape[0])
        self.sample_list += [slice_ndx for slice_ndx in range(index_count)]
        self.ct = ct

    def __len__(self):
        return len(self.sample_list)

    def __getitem__(self, ndx):
        slice_ndx = self.sample_list[ndx % len(self.sample_list)]
        return self.getitem_fullSlice(self.ct, slice_ndx)

    def getitem_fullSlice(self, ct, slice_ndx):
        ct_tensor = torch.zeros((self.contextSlices_count * 2 + 1, 512, 512))

        start_ndx = slice_ndx - self.contextSlices_count
        end_ndx = slice_ndx + self.contextSlices_count + 1
        for i, context_ndx in enumerate(range(start_ndx, end_ndx)):
            context_ndx = max(context_ndx, 0)
            context_ndx = min(context_ndx, ct.hunits_arr.shape[0] - 1)
            ct_tensor[i] = torch.from_numpy(
                ct.hunits_arr[context_ndx].astype(np.float32)
            )

        ct_tensor.clamp_(-1000, 1000)

        return ct_tensor, slice_ndx
```

As it is required to feed a given patient's CT slice by slice, a Dataset is built that loads a CT with a single series_uid and returns each slice, one per __getitem__ call. Two Dataset classes are defined: one acting as a general base class suitable for validation data, and one subclassing the base for the training set, with randomization and a cropped sample. The data produced will be two-dimensional CT slices with multiple channels.

The extra channels will hold adjacent slices of CT. For the segmentation model, each slice is treated as a single channel, and a multichannel 2D image is produced. For validation, one sample per slice of CT is produced that has an entry in the positive mask, for each validation CT. There are two different modes in which validation can be performed. First, using every slice in the CT for the dataset. This is useful to evaluate end-to-end performance. Second mode for validation during training, is limiting to only the CT slices that have a positive mask present.

5.3.3.2 Segmentation mask

```
def segmentCt(self, ct):
    with torch.no_grad():
        output_a = np.zeros_like(ct.hunits_arr, dtype=np.float32)
        seg_dl = self.initSegmentationDl(ct)  #  <3>
        for input_t, slice_ndx_list in seg_dl:

            input_g = input_t.to(self.device)
            prediction_g = self.seg_model(input_g)

            for i, slice_ndx in enumerate(slice_ndx_list):
                output_a[slice_ndx] = prediction_g[i].cpu().numpy()

        mask_a = output_a > 0.5
        mask_a = morphology.binary_erosion(mask_a, iterations=1)

    return mask_a
```

Segmentation is performed on every slice of the entire CT scan. The output is an array of per-pixel probabilities (that is, in the range 0…1) that the given pixel is part of a nodule. While iterating over the slices, the slice-wise predictions are collected in a mask array that has the same shape as the CT input. Afterward, the predictions are thresholded to get a binary array. A threshold of 0.5 is used. A small cleanup step using the erosion operation from scipy.ndimage.morphology is performed. It deletes one layer of boundary voxels and only keeps the inner ones—those for which all eight neighboring voxels in the axis direction are also flagged. This makes the flagged area smaller and causes very small components (smaller than 3 × 3 × 3 voxels) to vanish. Augmentation is done on the GPU. By moving the smaller input to the GPU first, the expanded data is kept local to the GPU, and less memory bandwidth is used.

Segmentation starts with the entire CT: hundreds of slices, or about 33 million (225) voxels (give or take quite a lot). About 220 voxels are flagged as being of interest; this is orders of magnitude smaller than the total input, which means 97% of the voxels are thrown out.

5.3.3.3 Grouping segmented candidates

```
def groupSegmentationOutput(self, ct, mask_a):
    candidateLabel_a, candidate_count = measurements.label(mask_a)
    centerIrc_list = measurements.center_of_mass(
        ct.hunits_arr.clip(-1000, 1000) + 1001,
        labels=candidateLabel_a,
        index=np.arange(1, candidate_count + 1),
    )

    candidates_list = []
    for i, center_irc in enumerate(centerIrc_list):
        center_xyz = irc2xyz(
            center_irc,
            ct.origin_xyz,
            ct.voxSize_xyz,
            ct.direction_arr,
        )
        assert np.all(np.isfinite(center_irc)), repr(
            ["irc", center_irc, i, candidate_count]
        )
        assert np.all(np.isfinite(center_xyz)), repr(["xyz", center_xyz])
        candidate_tuple = CandidateTuple(center_xyz)
        candidates_list.append(candidate_tuple)

    return candidates_list
```

A simple connected-components algorithm is used for grouping the suspected nodule voxels into chunks to feed into classification. This grouping approach labels connected components, which is accomplished by using scipy.ndimage.measurements.label. The label function takes all nonzero pixels that share an edge with another nonzero pixel and marks them as belonging to the same group. Since the output from the segmentation model has mostly blobs of highly adjacent pixels, this approach matches the data well. As output, a list of three arrays (one each for the index, row, and column) the same length as the candidate_count. This data is used to populate a list of candidateInfo_tup instances. As suitable data for the first four values (isNodule_bool, hasAnnotation_bool, isMal_bool, and diameter_mm) is not available, placeholder values of a suitable type are inserted.

Then coordinates are converted from voxels to physical coordinates in a loop, creating the list. While grouping doesn't remove anything explicitly, it does reduce the number of items, as consolidation of voxels into nodule candidates are done. The grouping produces about 1,000 candidates (210) from 1 million voxels. A nodule of 16 × 16 × 2 voxels would have a total of 210 voxels.

5.3.4 Classification

5.3.4.1 Luna dataset

```
class LunaDataset(Dataset):
    def __init__(self, ct, candidates_list):
        self.ct = ct
        self.candidates_list = copy.copy(candidates_list)

    def __len__(self):
        return len(self.candidates_list)

    def __getitem__(self, ndx):

        candidate_tuple = self.candidates_list[ndx]
        return self.sampleFromCandidate_tuple(candidate_tuple, self.ct)

    def sampleFromCandidate_tuple(self, candidate_tuple, ct):

        width_irc = (32, 48, 48)

        candidate_a, center_irc = ct.getCtChunk(
            candidate_tuple.center_xyz,
            width_irc,
        )
        candidate_t = torch.from_numpy(candidate_a).to(torch.float32)
        candidate_t = candidate_t.unsqueeze(0)

        return candidate_t, torch.tensor(center_irc)
```

The project will need to separate samples into a training set and a validation set. That is done by designating every tenth sample, specified by the val_stride parameter, as a member of the validation set. Another parameter isValSet_bool is accepted used to determine whether only the training data, the validation data, or everything should be included. A series_uid parameter can be passed, then the instance will only have nodules from that series. This can be useful for visualization or debugging, by making it easier to look at, for

instance, a single problematic CT scan. Two dataset classes are created for segregation between the training data and the validation data. This depends on there being a consistent sorted order to the candidateInfo_list, which is ensured by having there be a stable sorted order to the candidate info tuples, and by the getCandidateInfoList function sorting the list before returning it.

5.3.4.2 Classification (Nodule-Non-Nodule and Malignancy)

```
def classifyCandidates(self, ct, candidates_list):
    cls_dl = self.initClassificationDl(ct, candidates_list)
    classifications_list = []
    for _, batch_tup in enumerate(cls_dl):
        input_t, center_list = batch_tup

        input_g = input_t.to(self.device)
        with torch.no_grad():
            _, probability_nodule_g = self.cls_model(input_g)
            _, probability_mal_g = self.malignancy_model(input_g)

        zip_iter = zip(
            center_list,
            probability_nodule_g[:, 1].tolist(),
            probability_mal_g[:, 1].tolist(),
        )
        for center_irc, prob_nodule, prob_mal in zip_iter:
            center_xyz = irc2xyz(
                center_irc,
                direction_arr=ct.direction_arr,
                origin_xyz=ct.origin_xyz,
                voxSize_xyz=ct.voxSize_xyz,
            )
            center_irc = xyz2irc(
                [center_xyz.x, center_xyz.y, center_xyz.z],
                direction_arr=ct.direction_arr,
                origin_xyz=ct.origin_xyz,
                voxSize_xyz=ct.voxSize_xyz,
            )
            cls_tup = Cls_Tup(
                prob_nodule, prob_mal, center_xyz, center_irc
            )
            classifications_list.append(cls_tup)
    return classifications_list
```

Now that identified regions in the image that the segmentation model considers probable candidates are available these candidates are cropped from the CT and fed into the classification module. Since there is a candidateInfo_list from the previous section, so all that is to be done is make a DataSet from it, put it into a Data-Loader, and iterate over it. Column 1 of the probability predictions is the predicted probability that this is a nodule and is what needs to be stored. Output is collected from the entire group. The output probabilities are now subjected to a threshold to get a list of things the model thinks are actual nodules. In a practical setting, this output can be given to radiologist to inspect.

One way to quickly get results (and often also get by with much less data) is to start not from random initializations but from a network trained on some task with related data. This is called transfer learning or, when training only the last few layers, fine-tuning.

Features (the output of the last convolution) obtained by training the network on nodule versus non-nodule classification are not useful for malignancy detection. So, this process before training is going to cut out the last bit of the model and replace it with something new. In this case, a network that has been trained on similar data: the nodule classification network.

The following is required before malignancy classification:

1. Load the weights of the model to start with, except for the last linear layer, where the initialization has to be kept.
2. Disable gradients for the parameters that are not required to be trained. (everything except parameters with names starting with head).
3. Since validation set is used, a full set of annotations and malignancy information for each CT is available, which can be used to create a confusion matrix with the results.

5.4 Implementation Support

5.4.1 PyTorch

PyTorch is a library for Python programs that facilitates building deep learning projects. It emphasizes flexibility and allows deep learning models to be expressed in idiomatic Python. PyTorch’s clear syntax, streamlined API, and easy debugging make it an excellent choice for introducing deep learning. PyTorch provides accelerated computation using

(GPUs), yielding speedups in the range of 50x over doing the same calculation on a CPU. In this project, PyTorch is used to parallelize operations on the GPU using Tensors. PyTorch's Module, Dataset, DataLoader, torch.optim, etc, are few of the modules used.

5.4.2 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. To implement AI aspirations, you need to use a programming language that is stable, flexible, and has available tools. Python provides all of these, which is why it is suited for a project such as lung cancer detection. Python provides an extensive set of libraries such as glob, copy, pathlib, collections, numpy, scipy, torch, matplotlib, sitk etc which catered to the specific needs of this project. It also provides interactive programming which makes visualizations and debugging easier.

SimpleITK is an image analysis toolkit with a large number of components supporting general filtering operations, image segmentation and registration. It is built on top of the Insight Segmentation and Registration Toolkit ITK with the intent of providing a simplified interface to ITK. The SimpleITK library is used to read the byte data from the dataset and to extract a 3D numpy array from the image. Cypress library is available for testing.

5.4.3 JavaScript

JavaScript is a scripting or programming language that allows you to implement complex features on web pages, displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc. JavaScript was initially created to "make web pages alive". The programs in this language are called scripts. They can be written right in a web page's HTML and run automatically as the page loads. Scripts are provided and executed as plain text. They don't need special preparation or compilation to run. In this aspect, JavaScript is very different from another language called Java.

5.4.4 React

React is the most popular JavaScript front-end framework in use today. It's used by both established companies and new startups. React is a Javascript library for creating user interfaces. Here are three places you'll find it being used:

- Web development

- Mobile app development
- Desktop app development

5.4.5 Redux

Redux is an open-source JavaScript library for managing the application state. Redux is commonly used with libraries such as Angular or React to build user interfaces. It becomes difficult to manage the state of each component in the application when the size of the application becomes extremely large. It helps in updating and maintaining the state of each component in the application. Redux is nothing but an open-source javascript library that contains the state of the application. The working of Redux is very simple to understand. There are three building parts of the Redux as- Store, Actions, and Reducers.

5.4.6 AWS

Aws is an online platform that provides scalable and cost-effective cloud computing solutions. AWS is a broadly adopted cloud platform that offers several on-demand operations like compute power, database storage, content delivery, etc., to help corporates scale and grow. AWS S3 bucket is used to store the data (80GB), and AWS EC2 instance is created to use GPUs and CPUs with high processing power to speed up the training process. The Tesla V100 GPU with 16GB RAM is used for training the models.

5.4.7 Flask

Flask is one of the most popular python web frameworks because of its lightweight. Although it is micro it is an extensible python web framework. By providing the required functionality, flask accelerates the development of simple web application. So, Flask is more suitable for smaller, less complicated applications. Flask gives the developer varieties of choice when developing web applications, it provides you with tools, libraries, and mechanics that allow you to build a web application, but it will not enforce any dependencies or tell you how the project should look like.

Chapter 6

TESTING

6.1 Unit Testing

Unit testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

Table 6.1 Unit Testing

Test Case	Description	Expected Output	Actual Output	Result
1	Get candidates list from candidates.csv and annotations.csv files	Candidates list should be a list of CandidateTuple objects which is a named tuple with the following keys: isNodule, diameter_mm, series_uuid, and center_xyz	Candidates list generated is a list of CandidateTuple objects with the expected keys: isNodule, diamter_mm, series_uuid, and center_xyz	Pass
2	Check the data types of each key in the named tuple CandidateTuple	isNodule must be a boolean, diameter_mm must be a float, series_uuid is a string, and center_xyz must be a float tuple with length 3	isNodule is a boolean, diameter_mm is a float, series_uuid is a string, and center_xyz is a tuple of all floats with length 3	Pass
3	Load CT scan into a format which is usable by PyTorch	Read the metadata of the scan and convert the RAW image into a numpy array	The CT scan is loaded and converted into a numpy array	Pass

4	Check the data types of loaded Ct instance	hunits_arr must be a numpy array with shape (*, 512, 512), origin_xyz and voxSize_xyz must be float tuples of length 3, and direction_arr must be a numpy array with shape (3, 3)	hunits_arr is a numpy array with shape (*, 512, 512), origin_xyz and voxSize_xyz are float tuples of length 3, and direction_arr is a numpy array with shape (3, 3)	Pass

6.2 Integration Testing

Integration testing is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated.

Table 6.2 Intergration Testing

Test Case	Description	Expected Output	Actual Output	Result
1	Select RAW and MHD files to the input provided in the client	Should allow files with the extensions .raw and .mhd to be uploaded.	Allows files with the extensions .raw and .mhd.	Pass
2	Enable the button to upload the server	Once files with allowed extensions are selected to the input, the button to upload them to the server must be enabled.	Files with allowed extensions are selected to the input. The button to upload them to the server is enabled.	Pass
3	Upload the selected RAW and MHD files to the server	Should allow files with the extension .raw and .mhd. If files with the allowed extensions are provided, the server must decode them and must respond with 200 OK. Else, the server	Allows files with the extension .raw and .mhd. Files with the allowed extensions are decoded and stored in the data directory, and then the server responds with a 200 OK. For other files,	Pass

		must respond with a 400 BAD REQUEST.	the server responds with a 400 BAD REQUEST.	
4	Enable the button to predict results for uploaded CT data	Once the server responds with a 200 OK when the files are uploaded, the predict button must be enabled.	The server responds with a 200 OK when the files are uploaded, and the predict button is enabled.	Pass
5	Click predict button to generate the results	On clicking the predict button, the API to run the models on the uploaded data must be fired. While the models are run, the user should see a notification on the screen to wait until the results are generated.	On clicking the predict button, the API to run the models on the uploaded data is fired. While the models are run, the user sees a notification on the screen to wait until the results are generated.	Pass
6	Show prediction data on the Results page	Once the prediction data is generated, the server must respond with a 200 OK and the prediction results. Then the user must be redirected to the results page.	The prediction data is generated, and the server responds with a 200 OK and the prediction results. Then the user is redirected to the results page.	Pass
7	Render nodules plot on the Results page	On clicking the Plot Nodule button, the server must return the plots generated while predicting. Then the image must get rendered on the screen.	On clicking the Plot Nodule button, the server returns the plots generated while predicting. Then the image gets rendered on the screen.	Pass
8	Download nodule data from the server	On clicking the Download Nodule Data button, the server must return a zipped archive of the generated nodule data, and the client must be able to download it.	On clicking the Download Nodule Data button, the server returns a zipped archive of the generated nodule data, and the client is able to download it.	Pass

Chapter 7

RESULTS

Home Screen – Select Files

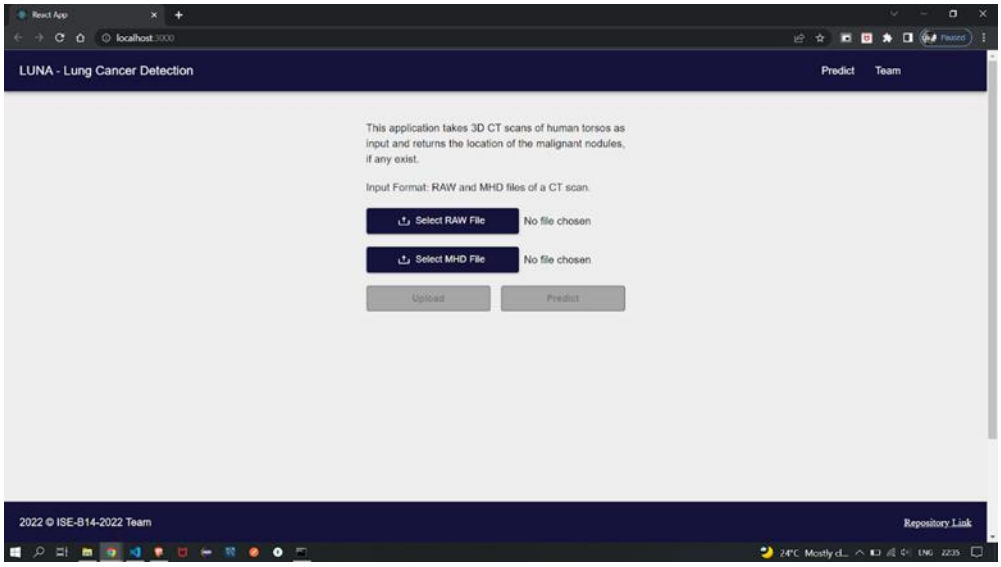


Fig 7.1 Home Screen 1

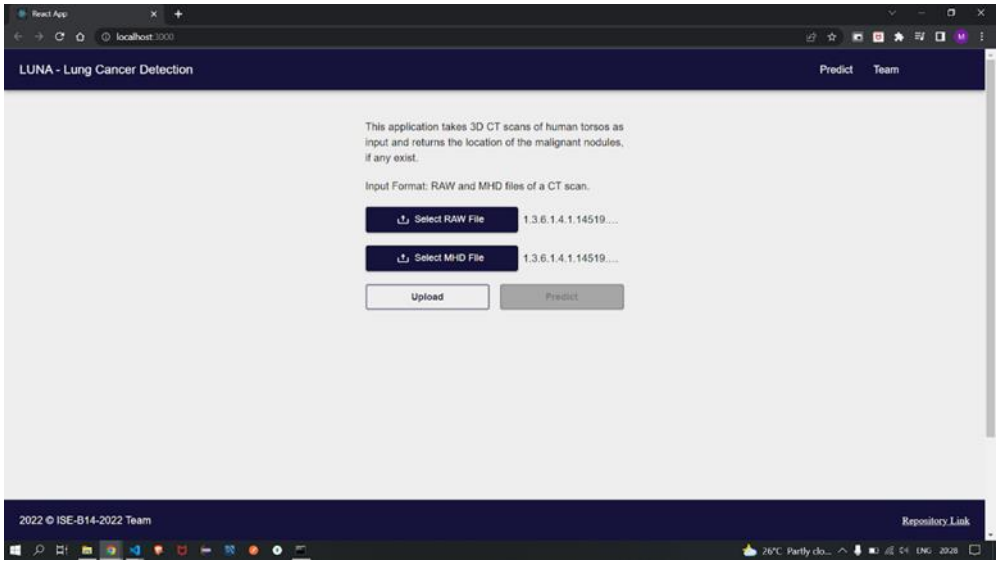


Fig 7.2 Home Screen with upload button enabled

When the web application loads, the user is provided with the following UI. The main feature of this application is that it allows users to upload CT scan data and predict the probability of malignancy in possible nodules. Until the user picks the required files, the Upload button will be disabled. The file extensions .raw and .mhd are allowed. Once the files have been chosen, the Upload button is enabled, and they can be uploaded to the server.

Home Screen – Upload Files to the Data Directory

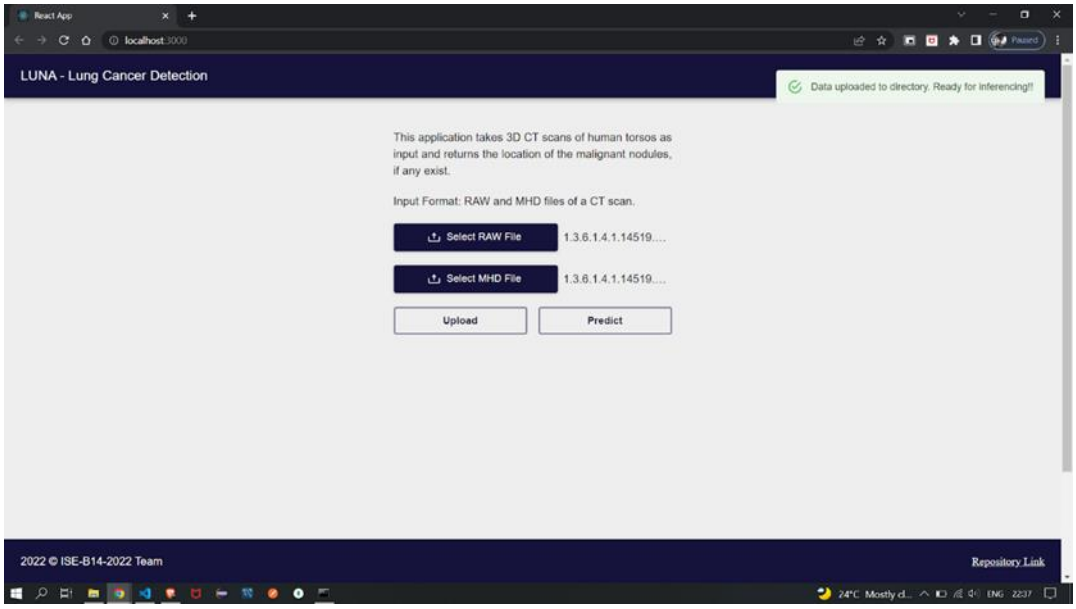


Fig 7.3 Pop up message for successfully uploading the files

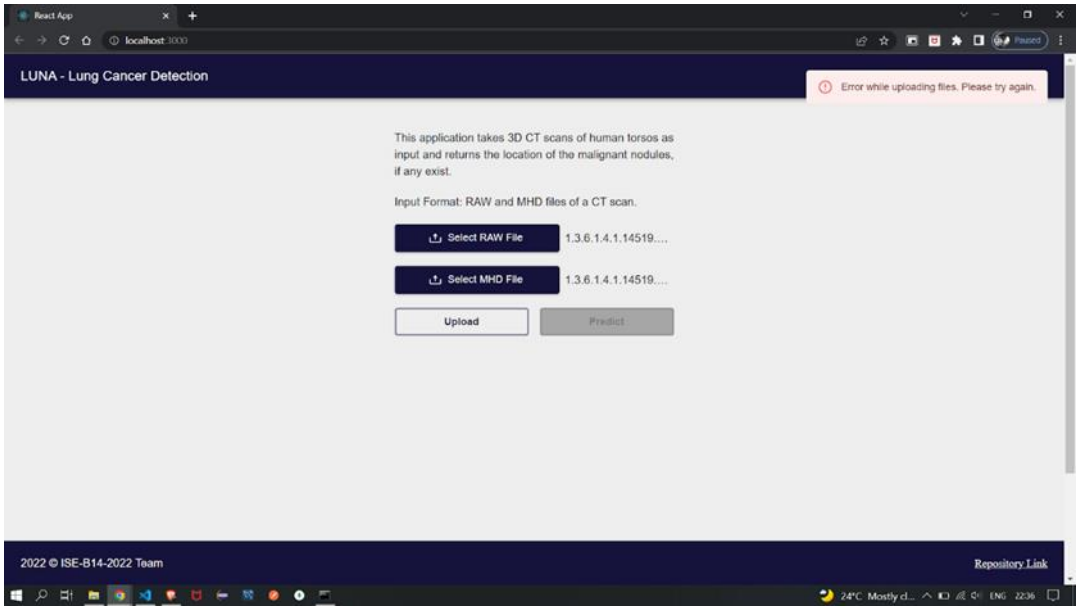


Fig 7.4 Error message on home screen

The /upload API is called when the user hits the Upload button to upload the selected files to the server. These files are decoded by the server and saved to the data directory. The server responds with a 200 OK if the upload process is successful; else, it fails. A message is sent to the user in either scenario. The Predict button is enabled once the upload is complete.

Home Screen – Predict

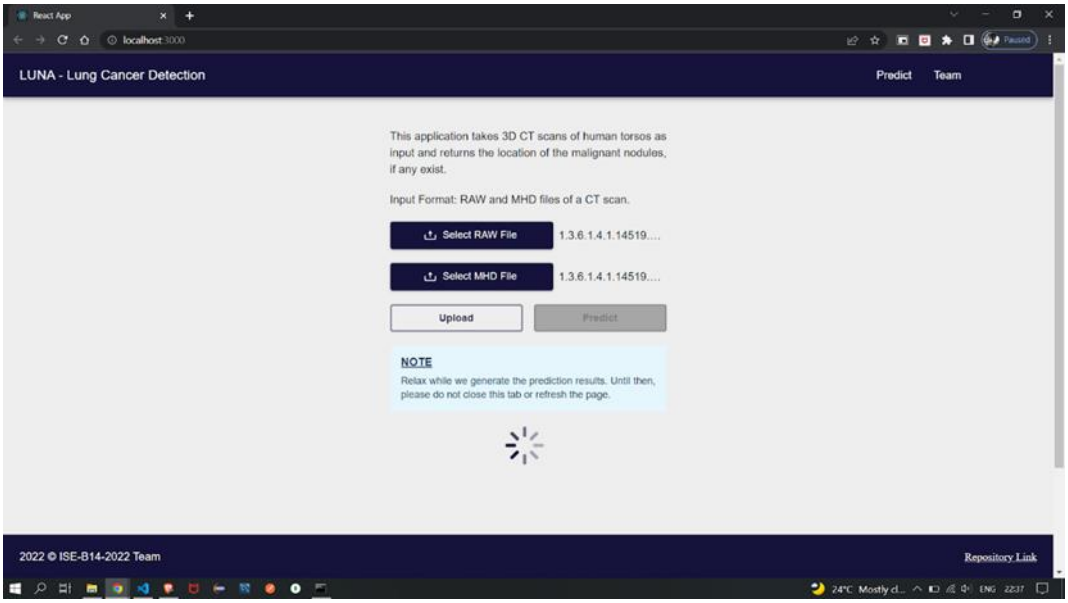


Fig 7.5 Page loading until files loaded in the data directory

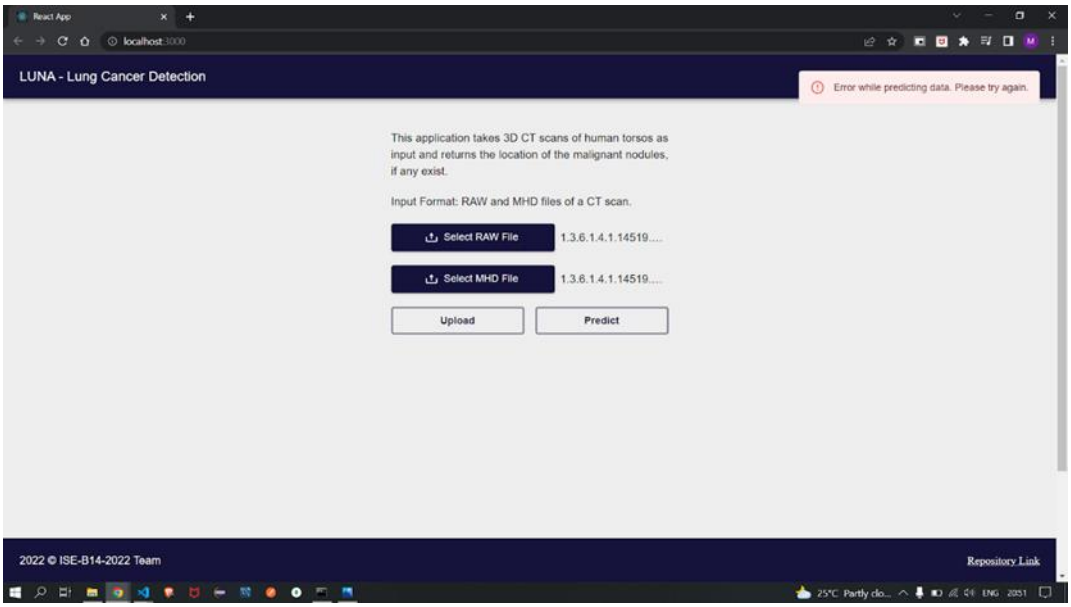


Fig 7.6 Predict button enabled

When a user clicks the Predict button, the /predict API is invoked. The server retrieves previously uploaded files from the data directory and runs the data through the prediction models. Because this takes some time, the user sees a message on the screen telling them to wait until the process is finished. Predict is disabled during the prediction process to prevent the user from triggering the action again. If a prediction error was made, the user is notified and given the opportunity to try again.

Results Screen

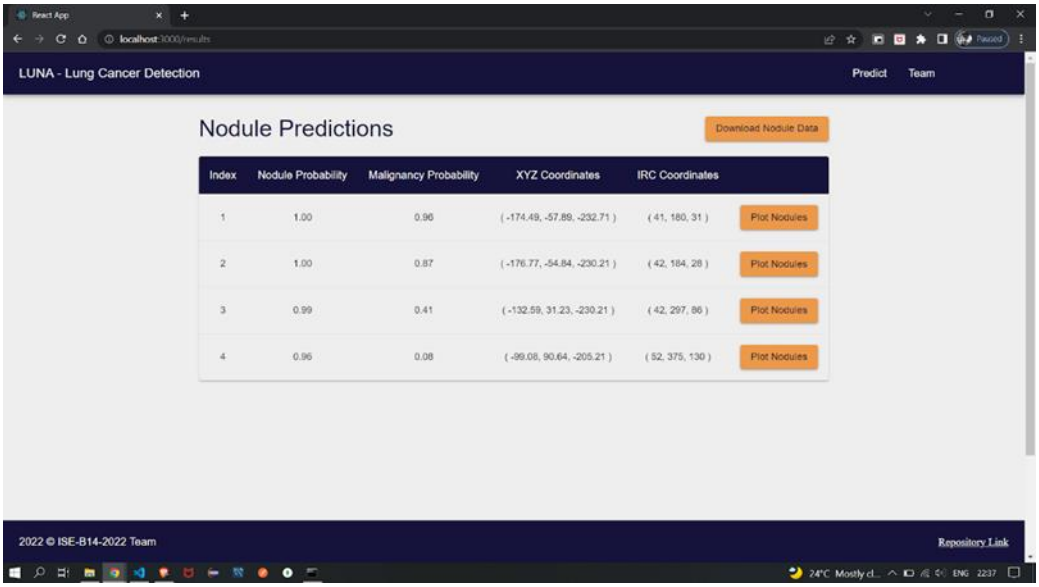


Fig 7.7 Nodule prediction

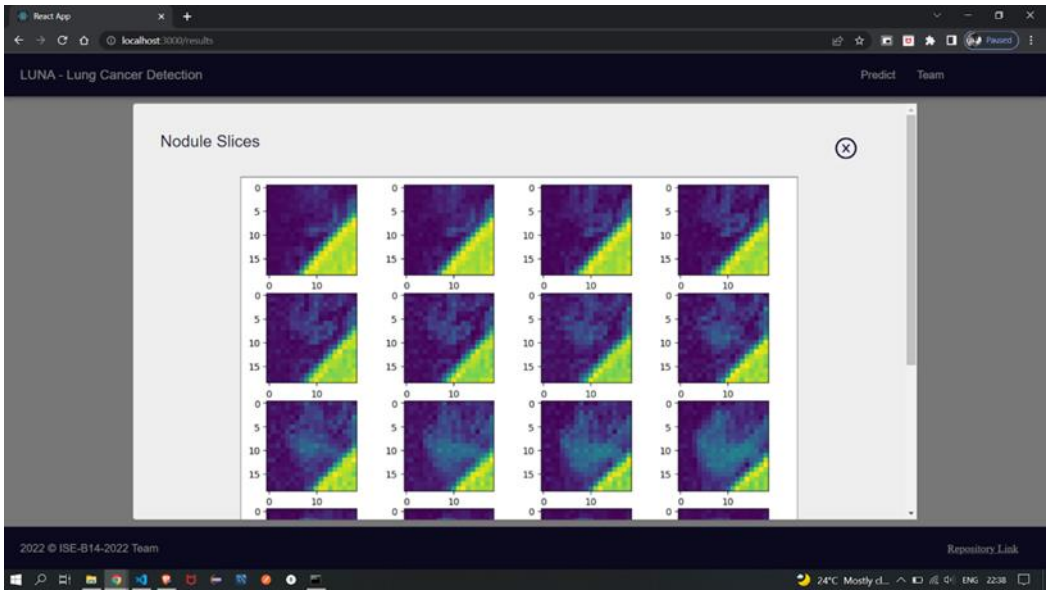


Fig 7.8 Nodule visualization

If the prediction is correct, the user is taken to the Results page, where the prediction data is presented in the form of a table. There is a Plot Nodules button attached with each possible nodule, which will call the /getplot API to get the nodule's image plots. The user may also be permitted to download the nodule data. This nodule data will be archived and transmitted in the same manner from the server.

Data Visualization

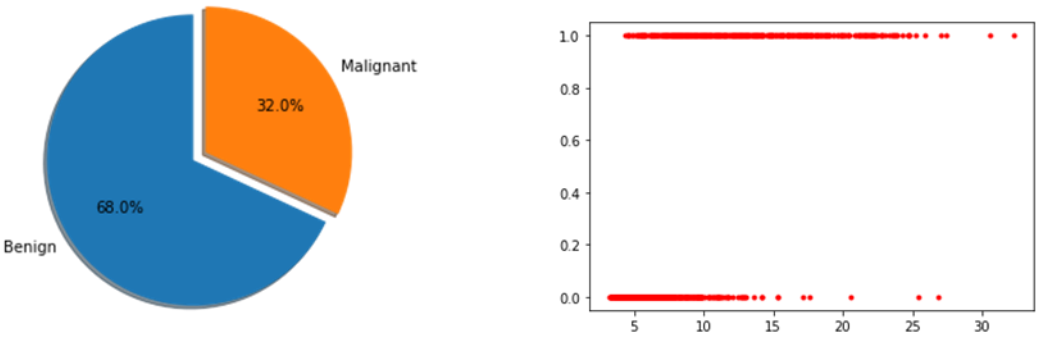


Fig 7.9 (a) Ratio of Malignancy vs Benign (b) Dependency of malignancy on nodule diameter

Data visualizations were performed on the LUNA annotations file to visualize the ratio between the number of malignant and benign nodules that have been manually flagged in the dataset as shown in Fig 7.9 (a).

The dependency of nodule malignancy to its diameter has been visualized as a scatter plot has been plotted. From the graph shown in figure 7.9(b), it can be seen that the nodules whose diameter are close to 0.1cm (1mm) can be concluded as benign. But a sure conclusion cannot be made about the nodules greater than 3mm. Hence it can be inferred that diameter alone cannot be used as parameter to decide the malignancy of a potential nodule.

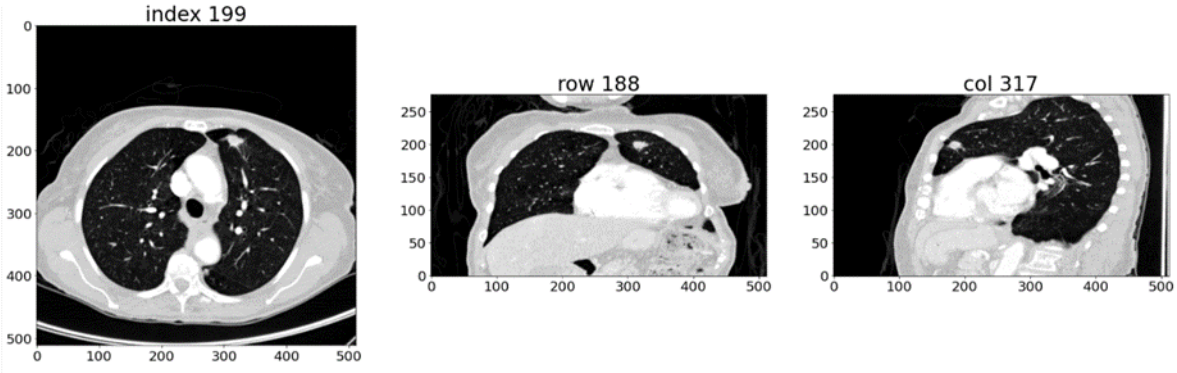


Fig 7.10 Visualization of three different angular views of the CT scan

Fig 7.10 shows a CT image as seen from three different views. The point is to get an intuitive sense of what the inputs to the automated system looks like.

Classification

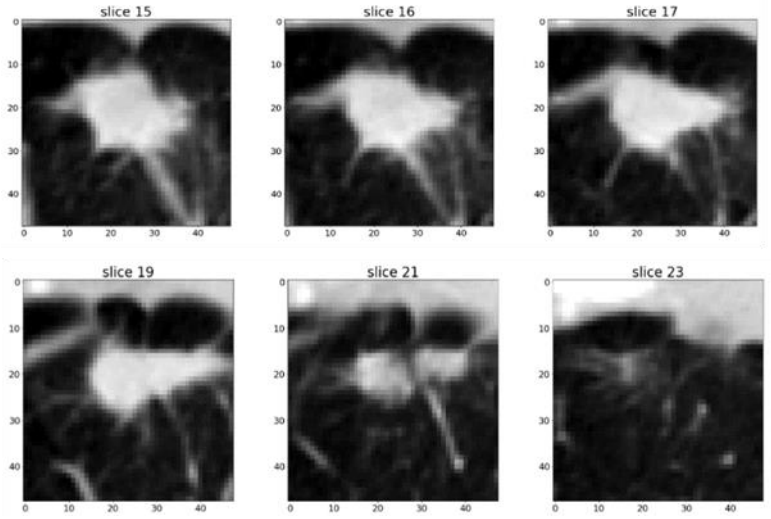


Fig 7.11 Slices of CT scan

The data that is passed to the initial nodule classifier are 3D crops of each nodule. Each slice gives a closer look on the orientation and spread of the nodule.

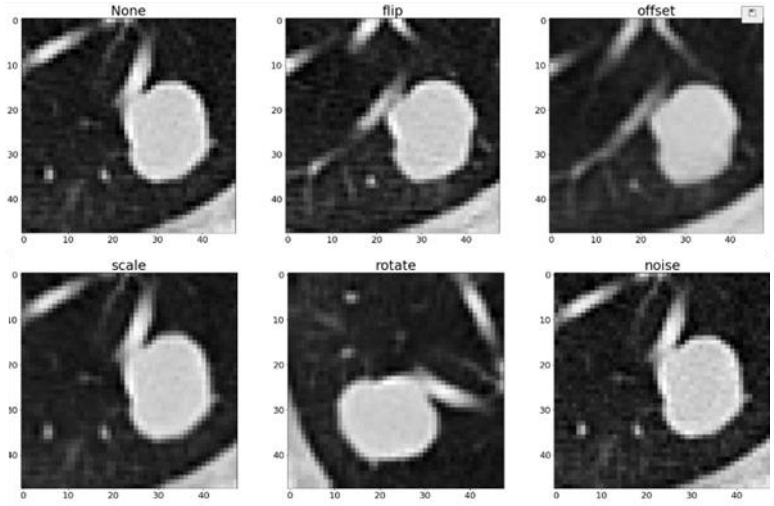


Fig 7.12 Augmented data

Data augmentation was performed on the existing 3D crops to increase the number of positive samples which was initially very less compared to the number of negative samples. Due to this reason the classifier did not produce the expected results of accuracy. Hence to bring about a balance in the data, 5 different techniques of data augmentation, namely, flip, shifting by offset, scaling, rotation and addition of noise were applied. Figure 7.12 shows the augmented slices of all the 5 techniques in comparison to a slice that has not been augmented.

Plotting Nodule Classification Training Metrics

The following different evaluation metrics have been used to determine the performance of the classification model:

1. Area under Curve (AUC) – The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes.
2. Recall – The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples
3. Correctness
4. Loss – It's a method of evaluating how well specific algorithm models the given data.

Table 7.1 Evaluation metrics of Nodule classification model

METRICS	TRAIN DATA	VALIDATION DATA
AUC	99.7	97.4
RECALL	0.99	0.92
CORRECTNESS	99.1	99.1
LOSS	0.02	0.06

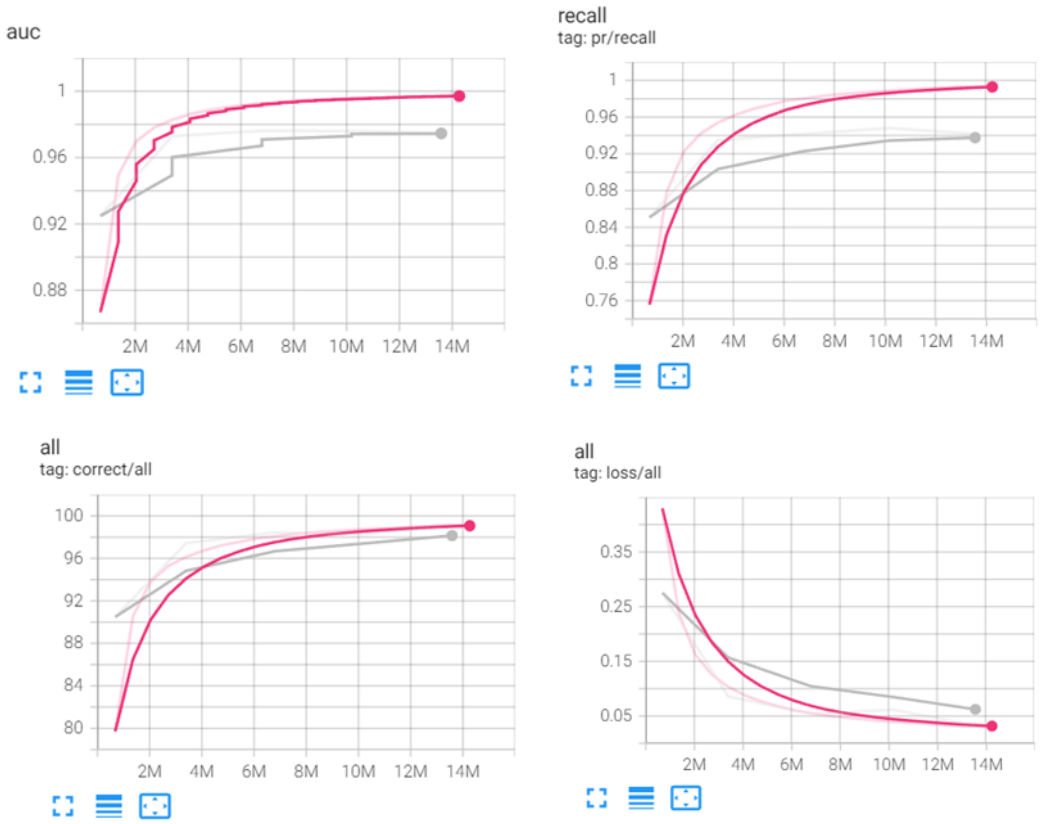


Fig 7.13 Classification Metrics: AUC, Recall, correctness, Loss

Segmentation

Initially data was prepared for the segmentation training. The 3D scans were converted to 2D images. Per-pixel masking was performed that produced these 2D images giving different pixel values for different regions of the CT scan. Per-pixel masking is used to filter out the dataset being used for training. As shown in Figure 7.14, different regions of interest have been masked for easier differentiation between the important regions and the surrounding regions. Lung_mask shows the entire lung region marked in blue. Negative and positive nodules have also been masked that helps in locating these nodules easily, as seen in the 2D plots of neg_mask and pos_mask respectively.

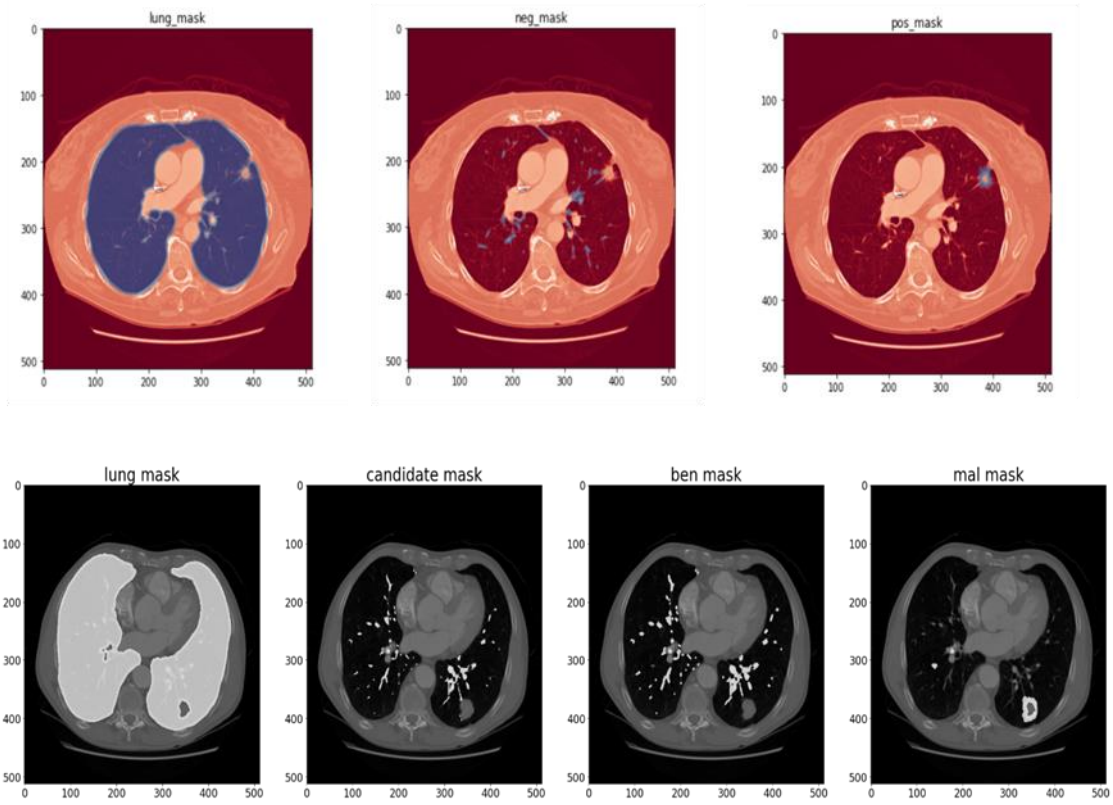


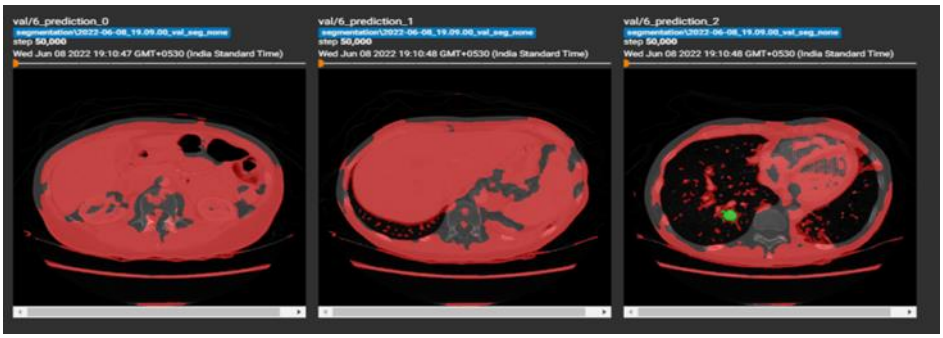
Fig 7.14 Segmentation Masks: Lung, Candidates, Benign, Malignant, Negative and Positive nodules

Results of Segmentation Training

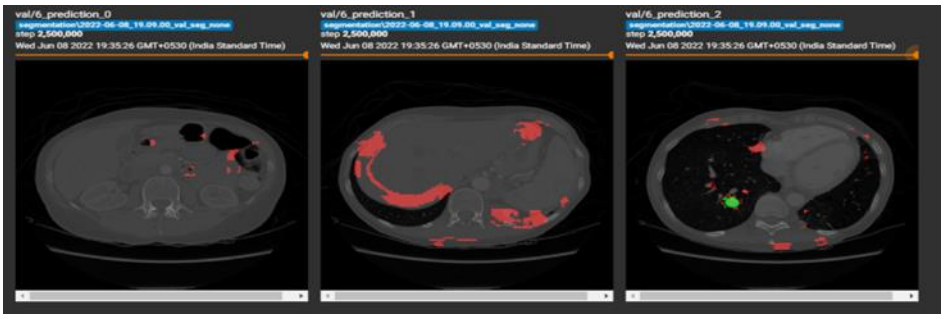
The following figure 7.15 shows the how the segmentation model slowly learns to segment and locate the possible nodules on both training and validation data. The green dot on the validation image indicated positive nodules.

(a) Mask on training data before Segmentation Training

(b) Mask on training data after Segmentation Training



(c) Mask on validation data before Segmentation Training



(d) Mask on validation data after Segmentation Training

Fig 7.15 Visualization of Segmentation training

Plotting Malignancy Classification Training Metrics

In addition to the evaluation metrics discussed before, another metrics called the Receiver Operator Characteristic (ROC) curve is used. ROC is an evaluation metric for binary classification problems. It is a probability curve that plots the TPR against FPR at various threshold values and essentially separates the 'signal' from the 'noise'.

Table 7.2 Evaluation metrics of Malignancy classification model

METRICS	TRAIN DATA	VALIDATION DATA
AUC	89.2	83.29
CORRECTNESS	84.6	77.27
LOSS	0.37	0.48

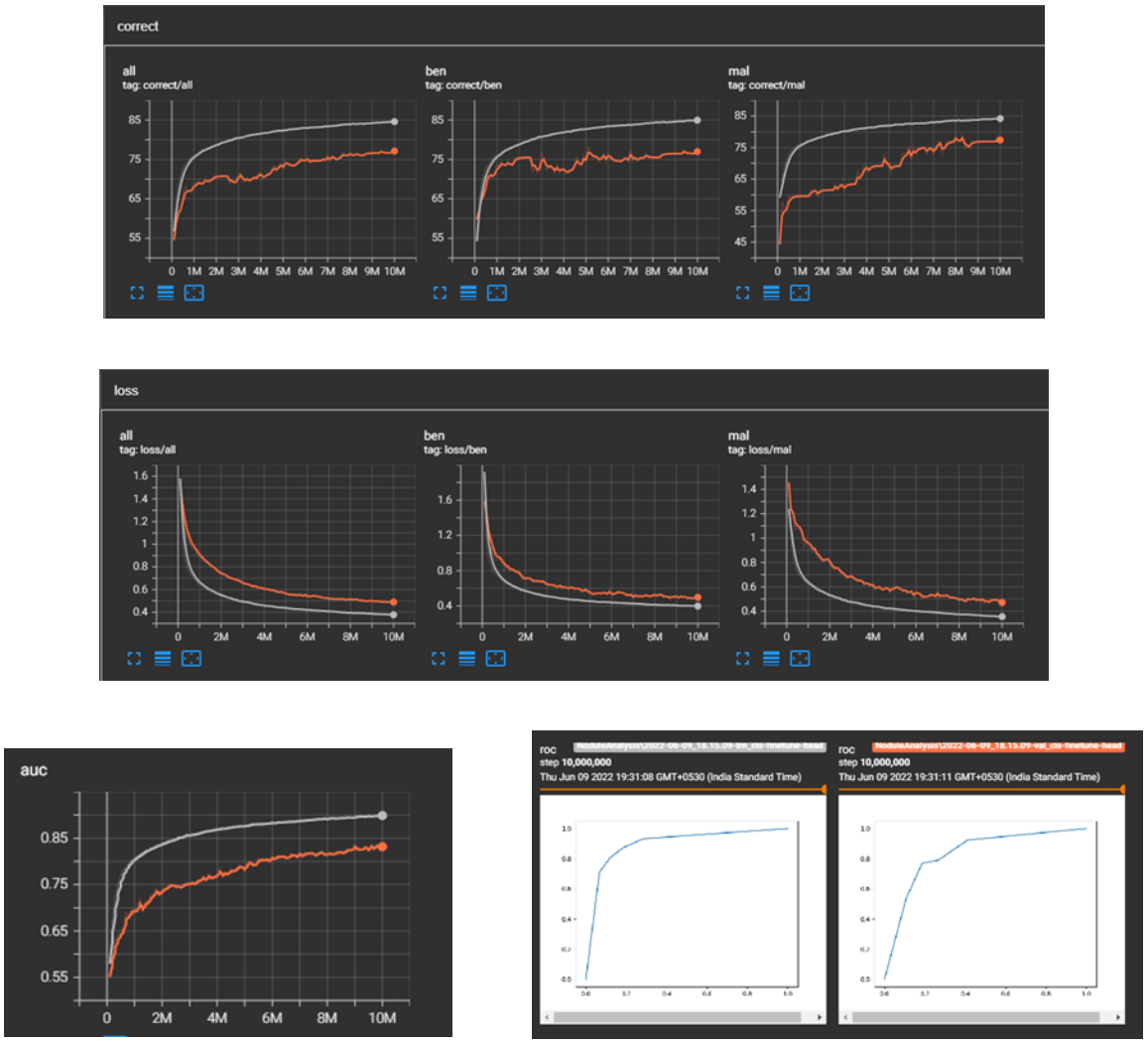


Fig 7.16 Malignancy classification Metrics: AUC, ROC, Correctness, Loss

Chapter 8

CONCLUSION AND FUTURE WORK

In this project, an automated lung cancer analysis system was implemented based on the image processing on the 3D CT scans of the lung. This system can help answer questions about the classification of cancer as benign or malignant. The lungs as seen on the initial CT scan processed with this system can assist in the field medicine to help diagnose lung cancer at earlier stages.

A neural network-based method is proposed to perform automatic lung cancer diagnosis. 3D Convolutional Neural Network (CNN) with a U-net backbone is designed to detect the nodules and a CNN classification model is used to evaluate the cancer probability of each detected nodule. Segmentation was performed on the CT scam images in order to distinguish potential nodules from the remaining lung mass. The overall system aims to build a working, end-to-end system that attempts to analyse lung cancer from CT scans and achieve good results on the cancer classification task. There is a distinction to be made between a fully automated system and one designed to supplement a human's talents. Once a piece of data is identified as irrelevant by the automated system, it is lost forever. However, when presenting data to a person, they should be allowed to peel back some of the layers and examine the near misses, as well as interpret the system’s conclusions with confidence.

REFERENCES

[1] Daniel Korat, "3D Neural Network for lung cancer risk prediction on CT volumes", 2020.

[2] Qurina Firdaus, Riyanto Sigit, Tri Harsono, Anwar Anwar, "Lung Cancer Detection Based On CT-Scan Images With Detection Features Using Gray Level Co Occurrence Matrix (GLCM) and Support Vector Machine (SVM) Methods", 2020.

[3] Ariful Hoque, A.K.M. Ashek Farabi, "Automated Detection of Lung Cancer Using CT Scan Images", 2020.

[4] Wadood Abdulg, "An Automatic Lung Cancer Detection and Classification (ALCDC) System Using Convolutional Neural Network", 13th International Conference on Develo Malignant Lung Nodule Detection using Deep Learning, 2020.

[5] Amrit Sreekumar, Karthika Rajan Nair, Sneha Sudheer, Ganesh Nayar H and Jyothisha J Nair, "Malignant Lung Nodule Detection using Deep Learning", International Conference on Communication and Signal Processing, July 2020.

[6] Jiachen Wang, Riqiang Gao, Yuankai Huo, Shunxing Bao, Yunxi Xiong, Sanja L. Antic, Travis J. Osterman, Pierre P. Massion, Bennett A. Landmana, "Lung Cancer Detection using Co-learning from Chest CT Images and Clinical Demographics", 2019.

[7] Nidhi S. Nadkarni, Sangam, " Detection of Lung Cancer using CT Scan Images", Borkar, the Third International Conference on Trends in Electronics and Informatics (ICOEI), 2019.

[8] Raunak Dey, Zhongjie Lu, and Yi Hong, "Diagnostic classification of lung nodules using 3D Neural Networks", 2018.

[9] Md Zahangir, Mahmudul Hasan, Chris Yakopci, "Recurrent Residual Convolutional Neural Network based on U-Net (R2U- Net) for Medical Image Segmentation", Feb 2018.

[10] Ruchita Tekade, Dr. K. Rajeswari, "Lung Cancer Detection and Classification using Deep Learning", Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018.

[11] Wentao Zhu, Chaochun Liu, Wei Fan, Xiaohui Xie, "DeepLung: Deep 3D Dual Path Nets for Automated Pulmonary Nodule Detection and Classification", 2018.

[12] Moffy Vas, Amita Dessai, "Lung cancer detection system using lung CT image Processing", 2017.

[13] Fangzhou Liao, Ming Liang, Zhe Li, Xiaolin Hu, Senior Member, IEEE and Sen Song, "Evaluate the Malignancy of Pulmonary Nodules Using the 3D Deep Leaky Noisy-or Network", 2017.

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", 2015.

[15] Aryan Mobiny, Pengyu Yuan,Pietro Antonio Cicalese,Carol Wu, "Memory-Augmented Capsule Network for Adaptable Lung Nodule Classification" 2021.

[16] Gopi Kasinathan and Selvakumar Jayakuma, "Cloud-Based Lung Tumor Detection and Stage Classification Using Deep Learning Techniques" 2022,

[17] https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5

[18] Semantic segmentation: https://www.jeremyjordan.me/semanticsegmentation/

[19] Cancer staging: https://www.cancer.gov/about-cancer/diagnosis-staging/staging

[20] LUNA Data set: https://luna16.grand-challenge.org/Data