PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

Report on

SEGEMENTATION AND PREDICTION FROM CT IMAGES FOR DETECTING LUNG CANCER

Submitted by

Vijetha K A (01FB16EEC334)

Vishwanath S (01FB16EEC343)

Rakshith Patil (01FB16EEC364)

January - April 2020

under the guidance of

Dr. Chethan K S

Designation: Assistant Professor

Department of Electronics and Communication Engineering

PES University

Bengaluru -560085

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGG

PROGRAM B.TECH

CERTIFICATE

This is to certify that the Report entitled

SEGEMENTATION AND PREDICTION FROM CT IMAGES FOR DETECTING LUNG CANCER

is a bonafide work carried out by

Vijetha K A (01FB16EEC334)

Vishwanath S (01FB16EEC343)

Rakshith V Patil (01FB16EEC364)

In partial fulfillment for the completion of 8th semester course work in the Program of Study B.Tech in Electronics and Communication Engineering, under rules and regulations of PES University, Bengaluru during the period Jan – Apr. 2020. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The report has been approved as it satisfies the 8th semester academic requirements in respect of project work.

Signature with date & Seal Dr. Chethan K S Internal Guide	*Signature with date & Seal Dr. Anuradha M Chairperson*	*Signature with date & Seal Dr. B. K. Keshavan Dean - Faculty of Engg. &Technology*

Name and signature of the examiners:

1.

2.

3.

DECLARATION

We, Vijetha K A, Vishwanath S, Rakshith V Patil, hereby declare that the report entitled, '*Segmentation and Prediction from CT Scan Images for Detecting Lung Cancer'*, is an original work done under the guidance *of Dr.Chethan K S, Assistant Professor, ECE Department* and is being submitted in partial fulfilment of the requirements for completion of 8th Semester course work in the Program of Study B.Tech, in Electronics and Communication Engineering.

PLACE: Bengaluru

DATE:

NAME AND SIGNATURE OF THE CANDIDATES

1.Vijetha K A

2.Vishwanath S

3.Rakshith V Patil

ACKNOWLEDGEMENT

We would like to sincerely thank to our guide and mentor **Dr.Chethan K S, Assistant Professor, Electronics and Communication Department**, PES University for his invaluable guidance, constant assistance, support , remarkable mentoring, endurance and constructive suggestions for the betterment of the project.

We would like to express our deep sense of gartitude to **Dr.Chandar T S, Prof. Swetha R,** and **Prof. Saritha Uniyal**. **Prof.Rajasekar M,** project coordinator**, Prof.Anuradha M , HOD ,** Electronics and Communication Department**, Dr.Sridhar,** Principal of **PES University, Dr. B.K.Keshavan , Dean of Faculty , Dr. Surya Prasad , Vice Chancellor, Prof. Jawahar Doreswamy , Pro Chancellor ,** and **Dr. Doreswamy , Chancellor** and **PES University** for giving us the opportunity to embark upon this topic and work on this project.

We would also like to thank all the teaching and non-teaching staff, family and friends for providing us the moral backing in realizing this project along with the facilities and conducive environment required for this project.

Thank you all.

ABSTRACT

Cancer is one of the most usual medical condition across the world. There are numerous types of cancer and lung cancer happens to be the most usual one amongst the population. Lung cancer affects people irrespective of their gender and is one of the most fatal medical condition in the wold. Identifying cancer at an early stage is a vital step that aids in minimizing the risk of death.

The CT scan data set of the lungs obtained from Kaggle and LUNA (Lung Nodule Analysis) websites has been implemented to perform classification of lung nodules.

In recent years, Deep learning and machine learning algorithms have been sought after to perform classification of lung nodules. From the Deep Learning library, machine learning and deep learning algorithms happen to be the most employed for the implementation of 3D Convolution Neural Networks and TensorFlow.

CONTENTS

LIST OF FIGURES

Chapter 1:

INTRODUCTION

1.1.Introduction

For machines the most complicated obstacle happens to be applications which involve sight and hearing. Unlike humans, for machines both the mentioned senses do not come naturally. But thanks to Deep learning and machine learning, the solutions to these limitations posed by machines are being implemented in a smooth and efficient manner and is only improving with every passing day. The invention of Artificial Neural Networks has been the biggest leap in the field of deep learning and machine learning. ANN, CNN and machine learning have been around for a while now, but only in theory and very rarely it has seen practical applications. One of the biggest challenges faced is that the technology has not been able to meet the demands and requirements of these algorithms in terms of processing power. But multiple layered structures have emerged (Deep learning) in today's era of fast paced growth of advanced technology.

In recent years, Deep learning techniques have been the most sough-after. There are many obstacles and challenges faced in the domain of image processing, signal processing and natural language processing and the go to solution to all these challenges has been the algorithms that come under 'Deep Learning'.

One domain where deep learning techniques have been the most sough-after is in the field of Biomedical Image Classification. There are numerous methods employed to identify ailments and for this purpose the most favoured preliminary step would be to have a look at Computed Tomography images(CT Images).

1.2. Objective

The outline of the project is to employee machine learning techniques and algorithms to analyse data (CT Scan slices) which have been already been pre-processed with numerous pre-processing techniques which include, keeping the slices of the DICOM image constant for all the patients, converting it into NumPy array and so on.

The objective was to identify and detect cancer at a very early stage in order to minimise the risk of death. In order to do this, we came across two major concepts in this domain,viz, Artificial Neural Networks and Convolution Neural Networks, and after excessive research we zeroed out on the 3D Convolution Neural Networks to train and validate a model of high performance and accuracy to label if a lung is cancerous or not. Hence, to build a model that would perform the rudimentary steps involved in lung cancer is the main outline of this project.

1.3.Problem Statement

Lung segmentation and classification of CT scan images for early detection of cancer.

Chapter 2:

LITERATURE SURVEY

[1]Qiu, Gnoping. "An improved recursive median filtering scheme for image processing." ***IEEE Transactions on Image Processing*** **5.4 (1996): 646-648.**

To perform the minimization of a two term cost function there are a wide range of optimization techniques available which will help us do the same but 'Median Filter' happens to be the most commonly used optimization technique employed. One major pro of using the median filter is that it provides a refined MSR(mean squared error)performance, unlike the already existing recursive filters. Images may be corrupted with pseudorandom, salt and pepper noise, Median filter happens to be the go to application that is implemented to reduce such noises that occur in an image.

The features of median filters are as follows

1. It is a non linear filtering technique
2. Sharpness is preserved in signals
3. Most effective in removing noise
4. Each pixel value is replaced with neighbouring value's median

Two dimension median operation can be taken forward in numerous ways. The first way to go about it would be by passing a 2D window over the 2D signal. The 2D window could be a cross or a square. The median value is taken as the output by ranking the points within the window.

A recursive median filter has been introduced for image processing in this paper.

A recursive median filter refines the MSE performance over the other already existing methods, this has been proved by the simulation results.

[2]Alam, Janee, Sabrina Alam, and Alamgir Hossan. "Multi-stage lung cancer detection and prediction using multi-class svm classifie." ***2018 International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2)*****. IEEE, 2018.**

The key factor that could help minimize the death rate associated lung cancer is by recognizing and identifying it at a very early stage and also by predicting if a pair of lung will develop caner or not in the near future. A very smooth and efficient manner of detecting lung cancer has been discussed in this paper by employing a SVM (Support Vector Machine).

For detecting lung cancer, a multi stage classification has been implemented. For segmentation threshold and marker controlled water shed algorithm has been implemented. The first step of the approach involves pre-processing the image with techniques such as applying a median and bilateral filter. The second step, it checks if the lung has been diagnosed with cancer or not, if a cancer cell or a nodule is detected, the algorithm classifies to which stage of cancer the lung is belongs to. The first, second, third or the final stage.

1) Image Enhancement: In order to carry out an image analysis, the image has to be pre-processed. Image enhancement is one of those techniques that pre processes the image to carry out further analysis. This analysis could be to plot a histogram or to model a CNN model, depending upon the application.
2) Image Segmentation: As the name says, image segmentation is a technique that enables us to partition an image(JPEG, DICOM, PNG, etc) into numerous segments. These segments collectively, when put together return the original image.
3) Feature Extraction: Feature extraction could be anything such as the size of the nodules detected in the lung region, size in terms of radius, length, volume, etc. For feature extraction a Grey Level Co-occurrence Matrix has been implemented.
4) Classification of cancer nodule: A support vector machine has been implemented for classification purpose. A ratio of the total lung area to the affected lung area is taken into consideration

For prediction a total of 200 images were taken into consideration and it predicted with an accuracy of 87% (174 images).

For detection, out of 100 images, it predicted that 3 lungs have cancer and the other 97 to be non-cancerous which gives it an accuracy of 97%.

[3]Sun, Shanhui, Christian Bauer, and Reinhard Beichel. "Automated 3-D segmentation of lungs with lung cancer in CT data using a novel robust active shape model approach." ***IEEE transactions on medical imaging*** **31, no. 2 (2011): 449-460.**

The methodology discussed is as follows, there are majorly two steps involved.

1) In order to outline the lung region an RASM (Robust Active Shape Model) is employed.
2) To segment the lung region an optimum surface finding approach has been implemented .

The following is included in the data set.

30 data sets, 40 cancerous lungs, 20 perfectly healthy lungs were evaluated to produce a result of 4mm in terms of MASDR (Mean Absolute Surface Distance Error)

Currently there are several limitations posed by the existing lung cancer detection models, the most common limitations being robustness and the relatively low processing speed. 'Region Growing Algorithm' is the fundamental idea behind the first idea while the second method uses 'Template Based Segmentation Approach'. In order to produce a segment of the infected lung area, from the CT scan image the ribs are identified to start the ASM matching. After employing ASM matching, the infected lung region is extracted.

Robust ASM Matching, as described by Rogers et al, this algorithm basically converts a segmentation problem into a graph optimization problem, the solution to which is obtained by the maximum-flow algorithm.

[4]Sun, Shanhui, Christian Bauer, and Reinhard Beichel. "Automated 3-D segmentation of lungs with lung cancer in CT data using a novel robust active shape model approach." ***IEEE transactions on medical imaging*** **31, no. 2 (2011): 449-460.**

There are majorly two approaches discussed in this paper and both the ideas are revolved around classifying the unique features by segmenting the pre processed images(DICOM CT Scan Lung Images) and then applying the Region of Interest.

The first approach is the 'Random Forest' approach while the second approach being 'Support Vector Machine' classification. With an efficiency of 94.5% the Support Vector Machine classifier proved to produce the best results.

Following are the steps involved

Grey scale conversion, Thresholding employed for binarization, Saliency Enhancement Technique for enhancement, watershed algorithm employed for segmentation of the digital image, Random forest and SVM for classification purposes, SURF algorithm for feature extraction.

Parameters such as mean ,median, variance, standard deviation helps us in determining if a lung is healthy or it is cancerous. The Support Vector Machine works in such a way that it categorizes the negative lungs and the positive cancerous lungs into two separate categories. With the help of more images and data set, the efficiency and performance of the model can be refined to a very large extent.

[5] Kalaivani, S., Pramit Chatterjee, Shikhar Juyal, and Rishi Gupta. "Lung cancer detection using digital image processing and artificial neural networks." In *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*, vol. 2, pp. 100-103. IEEE, 2017.

The main outline of this paper is to detect and diagnose lung cancer at a very early stage in order to minimise the risk of death by implementing an automated, efficient, smooth and a human error free model with the help of ANN model (Artificial Neural Network Model).
An efficiency of 78% was obtained after running for 25 epochs with a MSR of 0.1 in the validation.

Methods employed are as follows

1) Grey scale conversion of the image
2) Equalisation of the histogram
3) Thresholding employed for binarization
4) Extraction of features such as perimeter, major and minor axis, area and volume.
5) A three layer network, back propagation network

The system could perform without the presence of a radiologist if more data is used for training, testing and validation.

Chapter 3:

DATA SET INFORMATION

3.1.Data Set

The data set mainly consists of CT scan reports for 1600 patients. The complete data has been taken from two different sources which are Kaggle website and from the lung nodule analysis mainly known as luna. There is a file called DICOM which means digital image communications in the medical domain. All the DICOM files which are present, in that every person has his own DICOM file, it can be identified only with the help of a parameter called meta data which is mainly available in the DICOM file. The meta data is nothing but a unique ID which is given to every patient. This DICOM file mainly comprises of many slices of the entire chest region of that particular patient. All the slices which are present in the DICOM files are located at various geographic and geometric locations over spatial region and each slice is a 2D image, later from a 2D image it can be combined to form a 3D image of the entire chest region.

Fig 3.1.1 One 2-D Slice of The DICOM Image

DICOM Standard

DICOM is an acronym for Digital Imaging Communications in Medicine. Files in this for format are either stored with the extension DCM or DCM30 file extension, however some wont have an extension at all. It is used for both conversation protocol and a file layout

and a conversation protocol.

All the patients medical history, all his ultrasound scans, and the MRI photos are all clubbed into one file. This is done mainly so that the complete data can be transferred very easily between various devices where the DICOM format is supportable. There are two methods in which the files can be viewed which are:

1) With the help of RadiAnt FreeViewer
2) With the help of PyDicom

3.2. Opening the DICOM Images

1. Opening DICOM files with RadiAnt free viewer

A file viewer named radiant is used which is an open source software mainly for seeing the DICOM files. The meta data which is present with the image helps in making it smooth and can give access to the data

2. Using Libraries to extract the information

DICOM has its very own standard, even though there are many open source softwares and many open libraries, the package used here is PyDicom, this package is mainly used for making it work with the images in python.

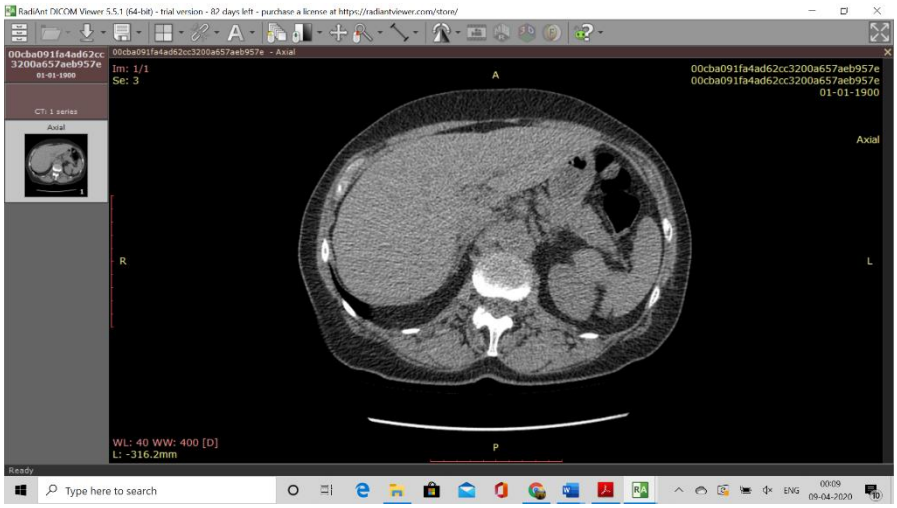

Fig 3.1.2 DICOM image opened with RadiAnt Free Viewer

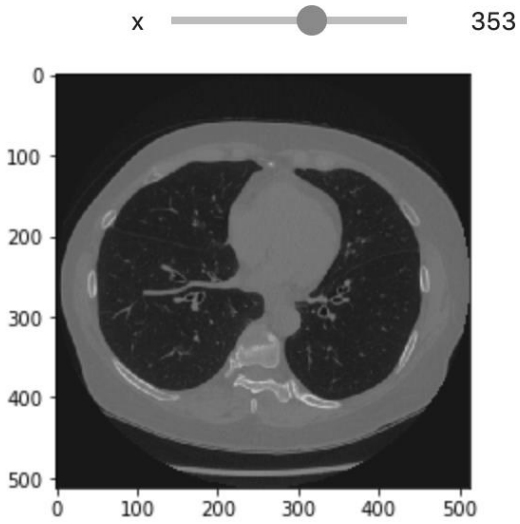

Fig 3.1.3 Image opened with PyDICOM

Chapter 4:

IMPLEMENTATION OVERVIEW

4.1.Introduction

- Anaconda Navigator – For python we have used jupyter notebook and we have to install anaconda navigator.
- NVidia CUDA – these tools are applied so that system performance can be way better then how it is performing before. It can be used only machines which are supported only by nvidia graphics
- to run the neural network on the jupyter notebook tensorflow fpu version is installed.
- For reading the dicom files pydicom has to be installed.
- For resizing or manipulating the images an opencv is installed.
- For visualization of images matplotlib is installed.
- To run the confusion matrix and to open and read csv files pandas are installed.
- For directory or file manipulations os is installed.
- To perform confusion matrix sci-kit learn is also installed.

4.2.Setup Process

After a task is being selected, after doing a very good understanding and research we have arrived at a conclusion of using this technology, and we wanted to ignore all the other technologies. So at first we had installed a anaconda navigator and along with that a python module has to be installed and a python version only 3.5 is supported, as it has all the required features for using and all those feature are again not present in the other versions. After all these being installed in our system later we even had installed nvidia cuda in the system, this software has mainly installed keeping in mind about the large sizes of the photos which we had to be working on and it takes up huge amount of time, if we are using the current CPU and hence here time is playing a very crucial factor. So for this reason we have used cuda, which consists of a gpu which can be used for processing the images and it also helps in analyzing the images so that it can help in processing of the dataset easily. There are many problems which have been faced while installing the cuda software because this software has many libraries and some of the have to placed in particular location and then only those files can run properly. This completely changed our mind to use

the tenser flow software and therefore importing the TFlearn was a very big challenge. Finally we were able to run the complete tensor flow with the dedicated modules which were available. There are many other libraries also which have been installed and used to complete the project and the description of those libraries are mentioned above.

4.2.1.Anaconda Navigator

Irrespective of what OS is being run on the system, Ubuntu, MacOS or Windows, in order to perform statistical analysis with R and Python language. Anaconda navigator has been the go to choice for the 20 million data science and machine learning community all across the globe. The following are the reasons why Anaconda Navigator happens to be a very popular choice

1. The wide range of multi-application packages that are easily downloadable.(10000+packages)
2. With the numerous libraries and packages that are available, this is bound to pose problems to manage the environment and variables. But Anaconda makes this entire process hassle free
3. The availability of packages to train, test and validate machine learning and deep learning models with TensorFlow and SciKit Learn.
4. Analyse huge volumes of data with packages such as NumPy, Pandas, Dask, Numba, etc with very high efficiency.
5. Packages such as the MatPlotLib, Datashader, HoloViews and Bokeh enables users to plot graphs, histograms, charts, tables helping users in visualising the data which or else could be a highly tedious task and prone to human error.

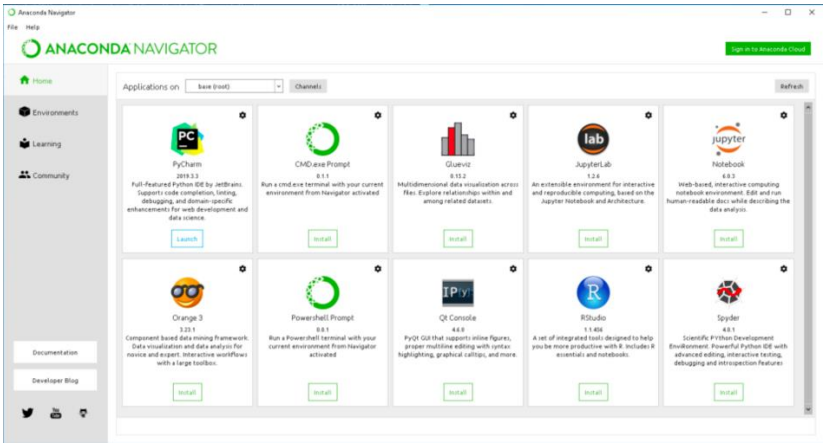

Fig 4.2.1.1 Anaconda Navigator

4.2.2.Numpy

When one talks about scientific computing in Python, NumPy is the first tool they would like to work with because of the following reasons.

1. Ability to create an array of dimensions N
2. Modern functions for broadcasting
3. Availability of tools that help integrate code which is written in C, CPP or even Fortran.
4. Perform operations which involve linear algebra, Fourier analysis (transform and series)
5. Perform matrix operations like inverse, multiplication, transpose or to even squeeze a matrix to the required size, to name a few.
6. Constantly keeps up to the demands and requirements of the users since it is an open source community and developers are periodically contributing to make it better.
7. Computation in NumPy is very efficient and time saving. NumPy has the capability to very easily club databases of different domains.

4.2.3.Pandas

With rapid advancements in technology, the capability to solve complex problems also grew thus giving birth to a community of developers independently working towards the goal of creating a data frame that is very scalable and easy to use thus leading to the development of a package named 'Pandas'. Pandas stands for Panel Data. Since it is an open source platform, constant updates are being made periodically. The advantages of using Pandas are as follows.

1. Easily create a data frame irrespective of the application and the domain. Be it something as complex as bio medical applications or something as simple as a data frame of the class marks.
2. Not all data sets are complete and pandas allows user to fill in the missing data with various techniques such as filling in the missing value with the average or even simply the missing row or column.
3. Enables users to work with file formats such as CSV and excel and even convert them into a pandas data frame in a very simple and efficient manner without hassle.

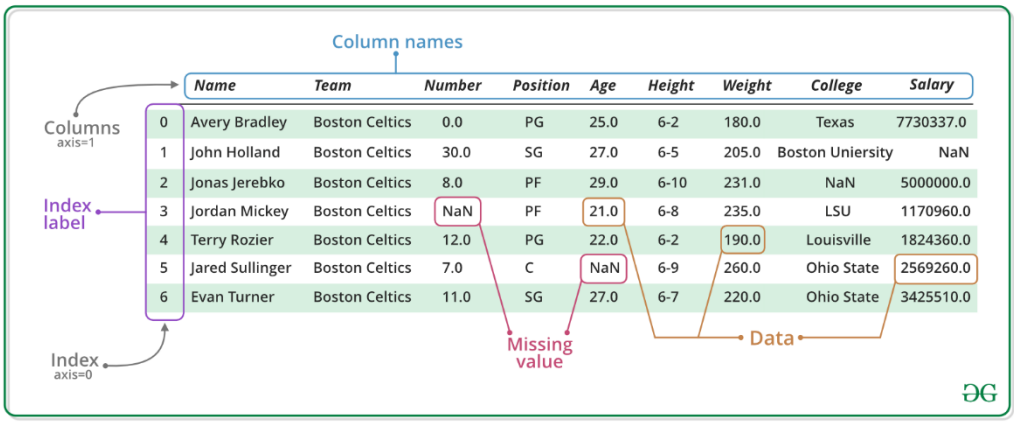

Fig 4.2.3.1 A typical Pandas DataFrame

4.2.4.SciKit Learn

SciKit Learn is the fundamental package for scientific computing with Python. It contains among other things:

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

4.2.4.SciKit Learn

For scientific or complex computing this is one of the most frequently used package.

SKlearn is also referred to as scikit learn

1) Huge amount of predictive analysis can be performed on the data were the users uses this scikit lean package
2) There are some algorithms which are available in this package which are knn and spectral, and clustering of the data cannot be performed in this package.
3) There are some parameters which are very important in considering an image or text file so that the data extraction from these are very easy. The attributes are also defined.
4) Scikit learn can be used only with license.

4.2.5.TensorFlor

With advancing technological improvements, more intricate problems came into picture that required robust machine and deep learning algorithms and this led to the birth of TensorFlow package. Deep learning algorithms can be easily implemented with the help of TensorFlow package in a very efficient manner to obtain accurate and desired results.

Following are the applications of TensoFlow

1. Voice/speech recognition
2. Sentiment analysis, typically done for Customer Relationship Management softwares.
3. Time series and Video detection.

4.2.6.OpenCV

Humans have the advantage if vision and perception while we hands down lack the computational power in comparison to machines. OpenCV aims at providing the advantage of vision and perception to machines enabling it to perform complex tasks which require both vision and the already existing computational and processing power. A combination of the best of both the worlds would enable a machine to solve very complex and intricate problems.

OpenCV stands for Open Source Computer Vision and the following are the features of OpenCV.

1. Read and write images
2. Detection of faces and its features
3. Text recognition in images
4. Modify and adjust image quality and colours
5. Develop augmented reality apps.

4.2.7.PyDICOM

PyDICOM is a pure package working with DICOM files. It was made for inspecting and modifying DICOM data in an easier pythonic way. The modifications can be written again in a new file.

As a pure python package, PyDICOM can run anywhere python runs without any other requirements, although NumPy is needed if manipulating pixel data.

What it can do:

- Read and write DICOM images
- Alter the pixels (if NumPy installed)

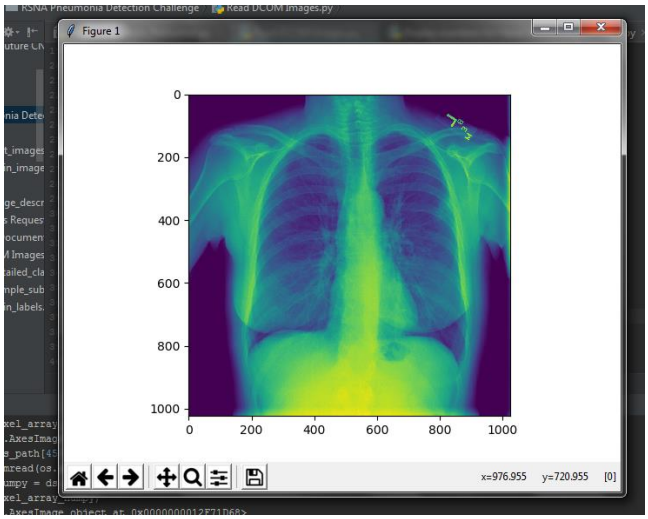

Fig 4.2.7.1 A DICOM image opened with PyDICOM

Chapter 5:

WORKING METHODOLOGY

5.1.Introduction

The working methodology basically consists of 5 steps :

1. Data pre processing
2. Data Visualisation
3. Segmentation
4. Computation of data via Neural Network using 3D CNN
5. Computation of confusion matrix

The working structure of the venture demanded gaining knowledge of concepts from CNN and going through study papers to get a comprehensive know-how of the activities involved in processing, training, and trying out a neural network.

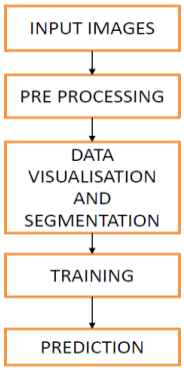

Fig 5.1.1 Block Diagram

5.2.Data pre-processing

Let's examine the data-set that we will be dealing with and to do so we will be incorporating RadiAnt DICOM to perform examination about the statistics. Therefore, with the above records we can apprehend what the data set consisted and the way we can use it to remedy that trouble statement. The patient ID call is provided for every affected person in each scan in the folder and ID acts as a unique identifier amongst the patients.

Taking more metadata tags into consideration we can find a few anomalies which might be the root cause for pre-processing. Example: Irregularities within the length of the x and y axis of the image, depth of the photo dependent on the quality of axial documents to be had within the folder, every so often it turned into the identical and occasionally not. The region of one single experiment of the entire 3D lung experiment was additionally also given for better information about which segment of the test we come dealing with.

The following steps were involved

1. Reducing pixel size and depth to process data
2. Sampling list of slices into chunks of list
3. Limiting each patient CT Scan to 20 slices
4. To process CT scans and save them as dimensional array for neural network
5. Saving the processed data as list

```
0
(20, 50, 50) [1 0]
(20, 50, 50) [1 0]
(20, 50, 50) [1 0]
(20, 50, 50) [0 1]
(20, 50, 50) [1 0]
(20, 50, 50) [1 0]
(20, 50, 50) [1 0]
(20, 50, 50) [1 0]
(20, 50, 50) [1 0]
(20, 50, 50) [1 0]
(20, 50, 50) [1 0]
(20, 50, 50)    This is unlabeled data
(20, 50, 50) [0 1]
(20, 50, 50) [1 0]
(20, 50, 50) [1 0]
(20, 50, 50) [1 0]
(20, 50, 50)    This is unlabeled data
(20, 50, 50) [1 0]
(20, 50, 50) [1 0]
(20, 50, 50) [0 1]
(20, 50, 50) [0 1]
(20, 50, 50) [1 0]
```

Output after converting the images to NumPy array

For code refer to Appendix [B] of page.50.

5.3. Data Visualisation

The following Python 3.0 modules were used

6. Numpy
7. Pandas
8. Pydicom
9. Scipy
10. matplotlib

The following steps were involved in the process of Data-Visualisation

1. Uplaoding the scans in the given folder path
2. Convert to Hounsfield Units (HU)
3. Combining 2D slices placed at different geometric locations to form a 3D image

Each section of a CT Scan image is associated with a particular value which is usually represented in Hounsfield Units.

The below chart will give us a better understanding of Hounsfield Units representation.

Substance	HU
Air	−1000
Lung	−500
Fat	−100 to −50
Water	0
CSF	15
Kidney	30
Blood	+30 to +45
Muscle	+10 to +40
Grey matter	+37 to +45
White matter	+20 to +30
Liver	+40 to +60
Soft Tissue, Contrast	+100 to +300
Bone	+700 (cancellous bone) to +3000 (dense bone)

Fig 5.3.1 Different Hounsfield Unit Values

With the help of matplotlib, we were able to plot a graph of 'frequency against Hounsfield Unit' for a particular patient's CT Scan image of the lung.

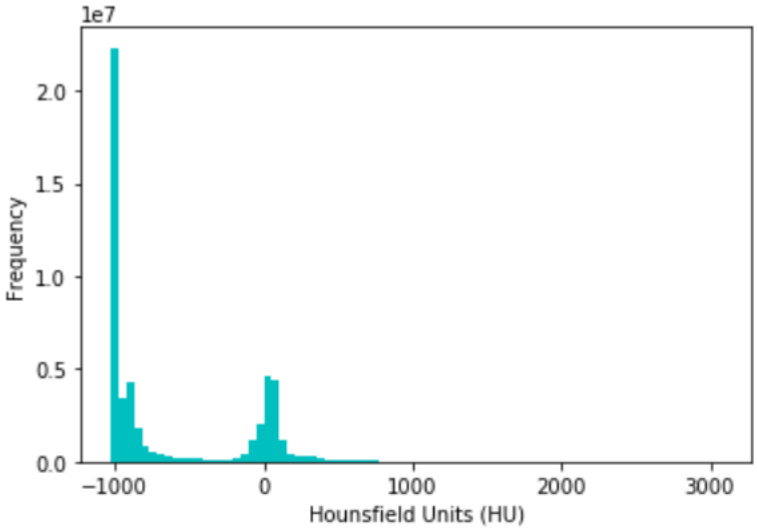

Fig 5.3.2 Graph of Frequency against HU

The next step of data visualisation involved compiling the different 2D slices of a patient's CT scan image placed at different geometric locations to form one 3D lung image.

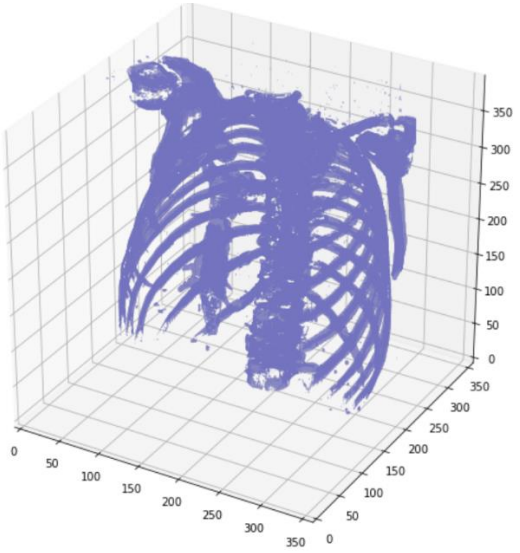

Fig 5.3.3 3D Image of the Chest Region

For code refer to Appendix[C] of page.51.
For code refer to Appendix[D] of page 52.

5.4. Lung Segmentation

5.4.1. Introduction

Before we go further with pre-processing it is important to eliminate those portions of the chest region that are not required and only retain the segmented portion of the lung region.

This has two advantages,

1. Reduces time and space complexity
2. Improves the efficiency of the model

5.4.2. Steps Involved

1. Since we know that the region around the lung is filled with air, a pixel around the lung Region would be labelled as ‘Air’.
2. Fill the lung structures with morphological closing operation.
   Morphological closing is a process where dilation is followed by erosion.
3. For every slice we determine the largest structure and this largest structure is labelled to be a portion of the lung.
4. Finally, all the scanned structures are put together to form the lung region.
5. This is converted into a binary image
6. The air packets are removed from the lung region (labelled as 0)

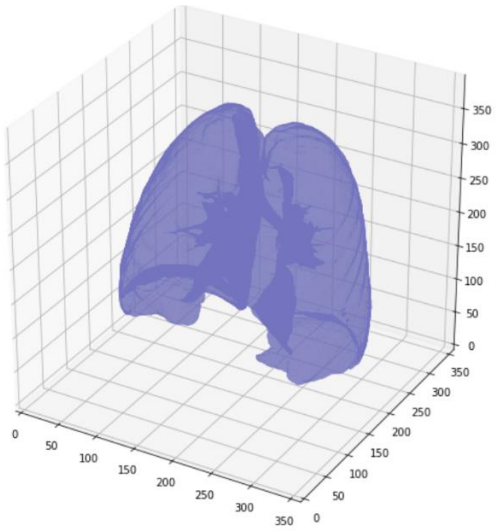

Fig 5.4.2.1 Segmented Lung Region
For code refer to Appendix[E] of page.53.

5.4.3.Dilation

The boundaries of objects in binary image is added with pixels to give it the perception of thickening or growth.

The number of pixels that are to be added depend on the size and shape of the SE(Structuring Element)

Let's say we have a cross shaped structuring element. This SE is moved across the image from left to right and from the top to bottom. Every time the structuring element overlaps with the pixel of an object in the binary image, it is called a hit and a pixel gets added around the pixel which has been just overlapped by the structuring element. This gives the perception of growth to an object in the binary image.

Mathematically speaking, the definition of Dilation is as follows for two sets X and B.

$$X \oplus B = \{x \mid -B_X \text{ Ç } X \text{ } ^1 \text{ } f\}$$

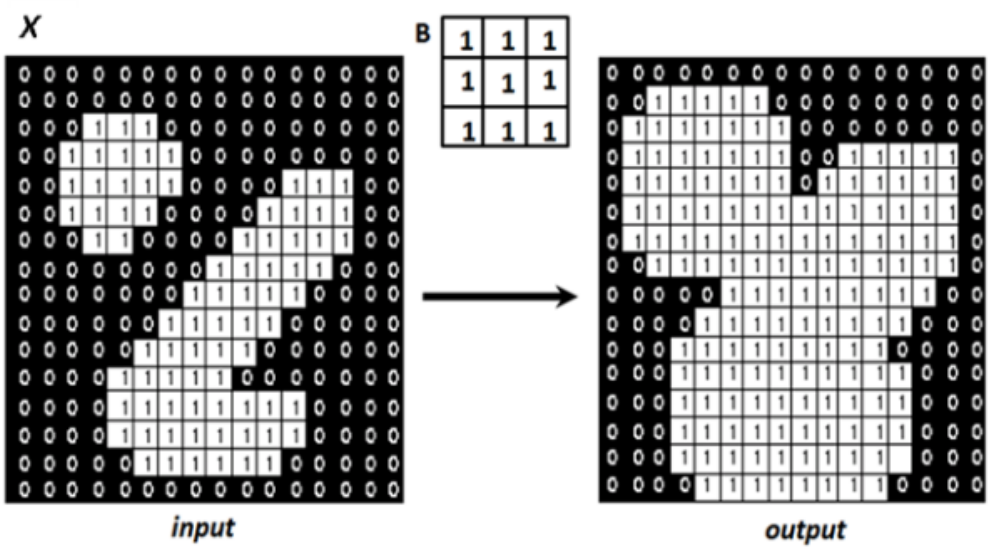

Fig 5.4.3.1 Dilation operation to bridge the gap with a 3 x 3 SE

5.4.3.Erosion

An object in the binary image sets the perception of shrinking after certain number of pixels have been eliminated around the boundary based on the size of the structuring element. Let's say we have a structuring element which is cross shaped. This structuring element is parsed from left to right and from top to bottom and every time there is a hit, the pixel of the object gets eliminated. This process is known as 'Erosion'

Mathematically speaking the definition of erosion for two sets X and B is as follows.

$$X \ominus B = \{x \mid B_x \acute{I} X\}$$

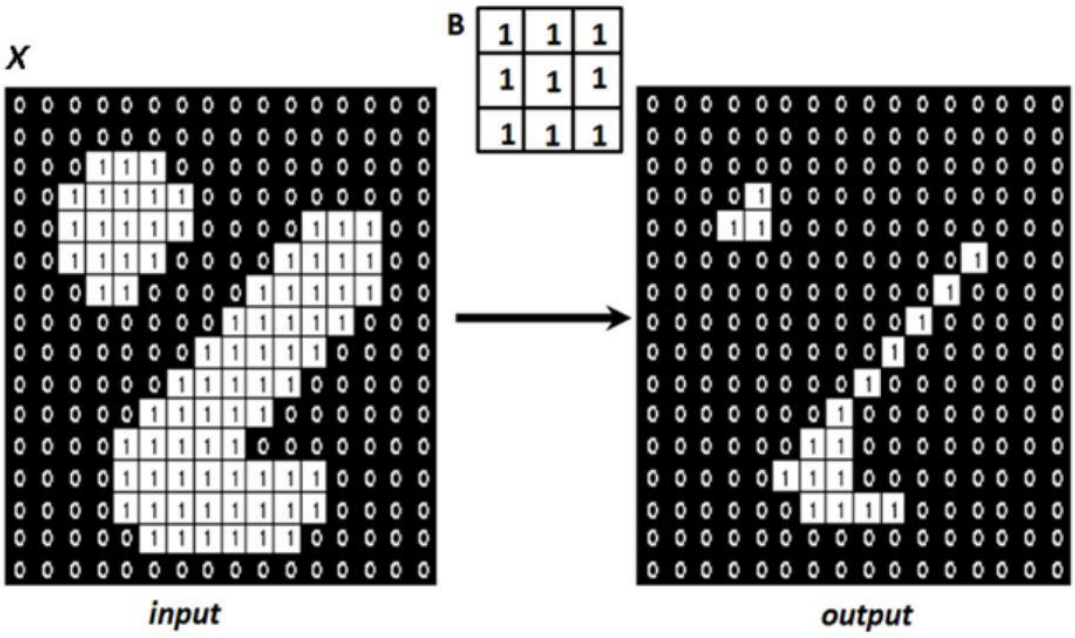

Fig 5.4.3.1: Erosion operation with 3 x 3 SE

.

5.5. 3D CNN Computation

5.5.1.Introduction

This section mainly comprises of the steps, which are required for pre-processing the data and the data which is obtained after pre-processing is used for the prediction. In this case the prediction is to tell whether a person will be suffered from cancer or not and the pre-processed data also helps in giving the accuracy of the model. To identify the amount of percentage which is being contributed to the convolution neural network and to even check what are the layers responsible for the processing of data. The description of each layer which is being used is given in the below convolution neural network model.

.

5.5.2.Convolution Neural Network Model

There are important and various operations which are being performed in the convolution neural network, so before we try to learn and understand about the UNET model, we need to have knowledge on the operations being performed in the neural network model.

A) Operations performed in Convolution

To perform a convolution operation there are two inputs given to it, which are:

1) An input image which should have a 3D volume and it should comprise a size of nin x nin channels.

2) There are some set of kernels present basically known as k filters and each one of them should comprise a size of f x f channels, where the value of the f can be either 3 or 5.

The output image which is obtained from the convolution operation, the output image is also a 3D volume which is same as the input image and it should comprise a size of nin x nout x k.

The connection between nin and nout is as described below.

$$N_{out} = \left[\frac{N_{in} + 2p - k}{s}\right] + 1$$

Nout = The number of input features that are present.

Nin = The number of output features that are present.

k = The size of the convolution kernel.

p = The size of the convolution padding

s = The size of the convolution stride.

For code refer to Appendix[F] of page.54.

The visualization of the convolution operations is shown below:

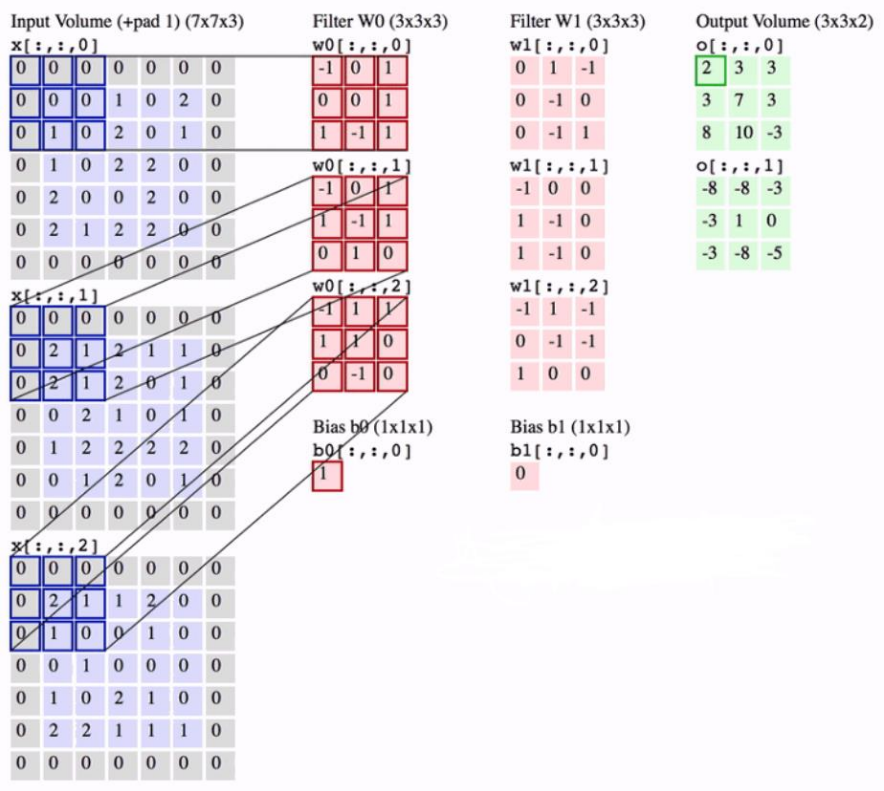

Fig 5.5.2.1 Visualisation of CNN

In the figure which is shown above, it has a input volume which has a size of 7x7x3. It also shows us that there are two filters each of them comprising a size of 3x3x3 and these filters also have padding size of 0 and a stride size of 2. So the output which has been obtained comprises a volume size of 3x3x2. If the understanding of the arithmetic is not up to the mark then all the concepts of convolution neural network should be revised, without the prior revision it cannot be moved further.

There is term called Receptive field which is being frequently used and is one of the most important terminology in this concept. This is nothing but it tells us that there is a filter it can be used a feature extractor to select some part of the region which is present in the input volume, it is shown in the above figure. The blue colour region which is highlighted in the input volume comprises a size of 3x3 which covers the filter at any particular instance this is the receptive field. This is sometimes it is even called as context.

To make it more simpler receptive field which is also known as context, it is the amount of the area which the filter covers the input image at any point of time.

B) The maximum pooling operation which can be performed

To explain better, to have very less parameters to be maintained in the network then the size of the feature map should as minimum as possible and this can be achieved only with the help of function known as pooling

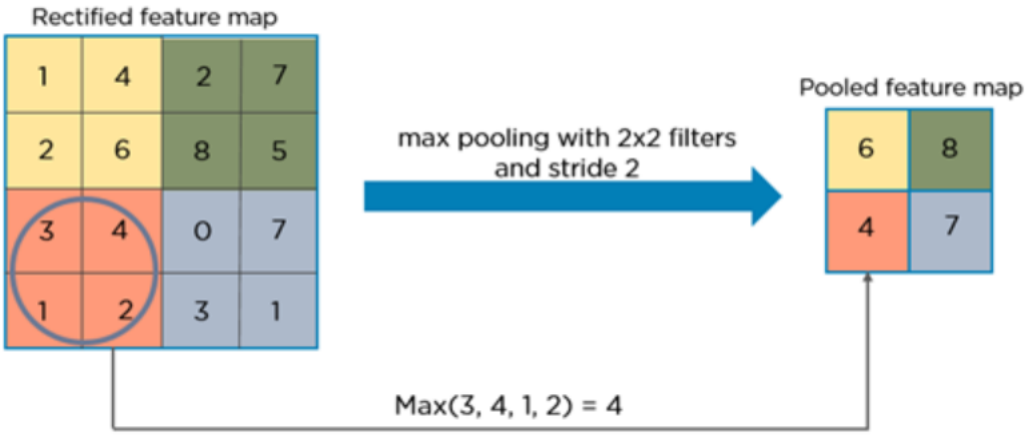

Fig 5.5.2.2 Max pooling operation

There are four different blocks in the above figure and the input feature map is present in each of the 2x2 blocks, a pooled featured map can be gained by maximizing the pixel value. To maintain within the limit of the maximum pooling operation the dimensions of the critical parameters should be within the max pooling operation and the two parameters which a meant to be critical are the filter and the strides. The main concept behind the max pooling area is that those critical parameters as mentioned above can be maintained easily from each area and the fact is that those parameters are no more meant to be critical. This information gives the best explanation of the context which is present in the above image.

The main factor to be taken into consideration is that the size of the images can be completely reduced with the help of convolution operation and pooling operation. This can also related to down sampling. In the example which is shown above before pooling the scale of the photograph 4x4 and the same photograph after pooling is 2x2. When it comes to reality which is nothing but down sampling means changing the resolution of the image from high to low. So therefore before pooling whatever data was present in a 4x4 photo, after pooling also it almost the data remains the same but in a 2x2 photo. Now if convolution operation is performed again, then the filters which are present inside the layer will be able to see a large content at a time. As we go more into the network the picture dimensions reduce no matter what the receptive subject keeps increasing.

C) Up-sampling

To explain it in easier words, the main aim of pooling is to reduce the dimension of the featured map, so that whenever we are getting inside the network there will be only few to be taken care.

As mentioned previously, the output which comes from the semantic segmentation it;s just not a class label or the parameters which are in the boundary. The output which has been obtained from the semantic segmentation is completely a high-resolution image in which all the pixels of that particular image can be classified.

So if we use a general convolution neural network which consists of two different layers which are the pooling layers and the dense layers, if this is being used then we will completely loose the where data and we just remain with what data, which is not at all required in this process. When segmentation comes into picture we need to have both the what data as well as the where data.

Hence a sampling needs to be done on the image, i.e. to recover the lost where data the image should now be converted from a low-resolution image to a high resolution image.

In the literature survey which has been done, there are many number of techniques for sampling a particular image. Some of them can be bi-linear interpolation, cubic interpolation and many more. However in most of the cases a transposed convolution is used for sampling a particular image.

D) Transported Convolution

Sometimes the transported convolution is also referred to as deconvolution or even fractionally strided convolution. This transported convolution is a method for sampling the image with the parameters. As the level keeps increasing then the transported convolution becomes less effective and then the regular convolution comes into the picture i.e. if the resolution of the image is high in the output quality and it has a low resolution picture in the input volume.

The complete system can just be reversed by just taking the transpose of the filter matrix.

5.6. UNET Architecture

5.6.1.Introduction

The UNET model was mainly developed for the segmentation of the bio medical images. The architecture which is present in UNET mainly consists of two paths. The first path in the architecture is the path contraction which is also known as the encoder. This is mainly used to reduce the context which is present in the image. The encoder has a combination of two different layers which are the convolutional layer and the max pooling layer. The second part in the architecture is expanding its path symmetrically which is also known as decoder, with the help of the transposed convolutions it enables the usage of certain specific locations. So therefore it is a fully convolutional network. It only uses convolutional layers and dense layers are not being used so that it can accept any image of its own size.

.

.

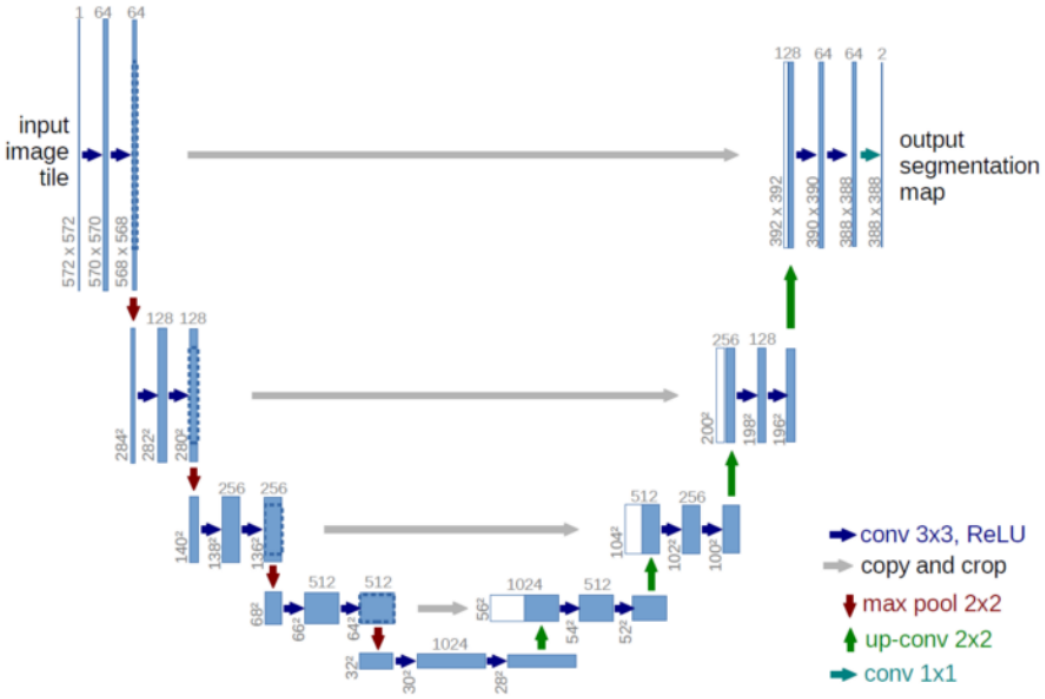

Fig 5.6.1.1 UNET CNN Model

The blue box in the above figure refers to the multi channeled featured map. In the figure is a box at the top which indicates the number of channels. At the lower end of the box the x-y size is mentioned. The boxes which show the copied featured maps are represented by white color. The different operation performed are denoted by arrows.

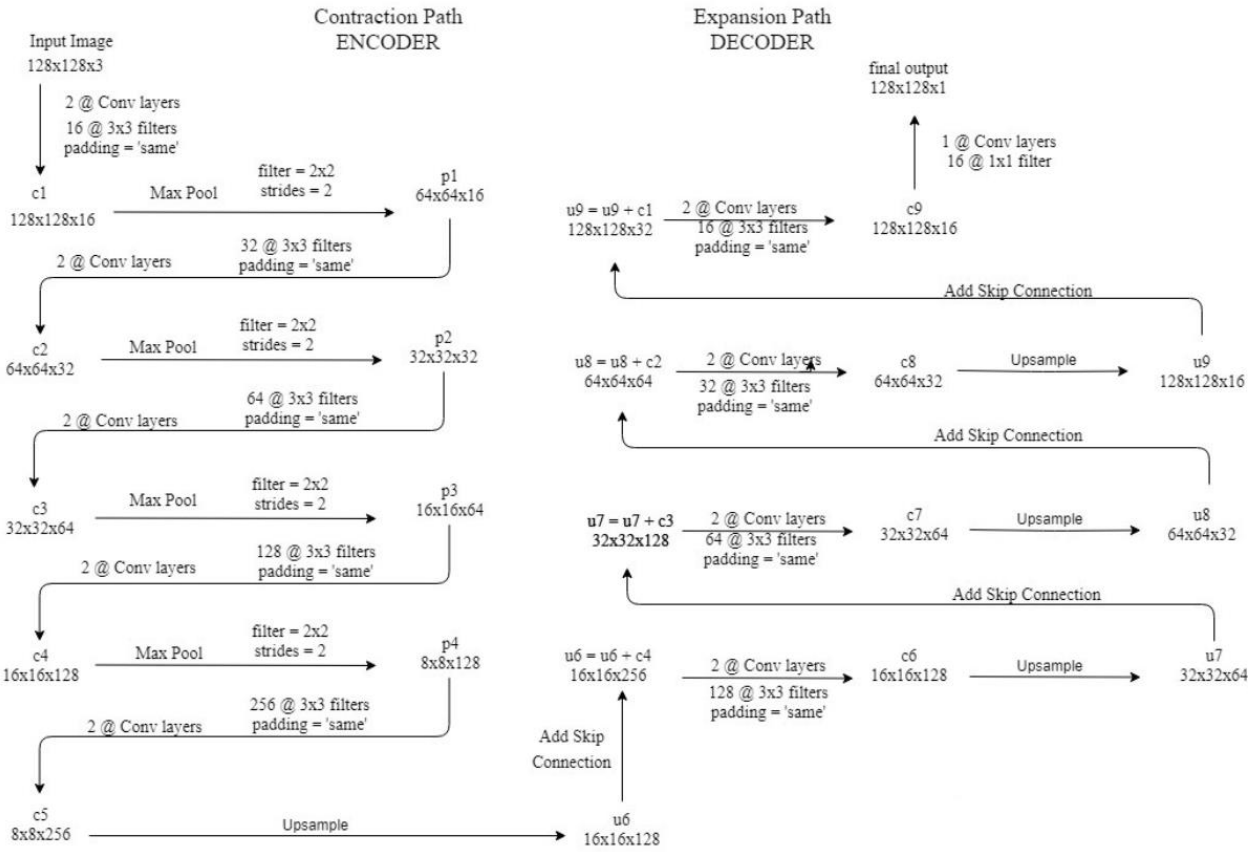

Fig 5.6.1.2 UNET Diagram

1. In the above flow diagram 2@conv layers tells us that there are two more layers which are being used.

2. The convolutional layers have output tensors which are labelled as c1,c2…..,c9.

3. The max pooling layers have output tensors which are labelled as p1,p2,p3.

4. The up-sampling layers also have output tensors labelled as u6,u7,u8.

5. The contraction path is present on the left hand side.

6. The expansion path is present on the right hand side .

7. As the depth increases the size of the images also decreases this can be done only with the encoder, where the size varies from 128x128x3 to 8x8x256.

8. As the depth decreases the size of the images also increases this can be done only with the decoder, which the size varies from 8x8x256 to 128x128x1.

9. u6 = u6 + c4

   u7 = u7 + c3

   u8 = u8 + c2

   u9 = u9 + c1.

The term UNET is derived from the above relation, the above relation gives itself a U shape.

5.7. Training

5.7.1.Introduction

The inputs are given in the form of 3D images, which are being pre-processed and there some sort of values which are being set within the diverse layers. There is no fixed rule that the values does not remain the same at all the point of time. So for getting the certain value there are multiple tests which have to be done frequently so that a particular value can be obtained and the value stays on the top. So basically when a model is being trained the information is automatically being split so that only a part of the information is used for training the model and the remaining information is just preserved for the testing process no matter even if the model is trained or not. We have almost trained the model with 1600 images and saved around hundred images for the testing purpose.

The main idea here is to complete the training of the model with 1495 images and pass the next 100 testing images to check how much accuracy is shown in other words to tell how well the model has been trained and to give the accurate results. To be more precise we are just giving the testing data as the input to the trained model to just check the accuracy.

The range is being set for the batch size so that the epochs can run, and whenever input data is given it specifies that how many images have to be processed at only once. The dimensions of the input size are also restricted to 50x50x20 but the machines which we are training can only adjust itself to certain capacity. So because of this reason the data sets quality and quantity has been completely reduced. We have noticed certain drops in the metrics, the main reason for the drop is that there are no sufficient records for the model to be trained, and it also affects the accuracy of the model adversely, when the model ran for 100 epochs we had made an observation that the accuracy of the model remained same at that 83rd epoch and we have obtained an accuracy of 0.70, which has been maximum accuracy so far for this model. We could not read more, as there has been a huge data loss causing the model to completely overfit the model. Before finding the final accuracy of the model we have tried mix and match between some of the layers so that the accurate composition can be obtained. To be more precise first we has set three layers and executed it we have got an accuracy of 54% with just 3 runs of epochs and we have observed that the accuracy has been improved. We saved the previously used layers and extended that to six layers which has become very much efficient and it just changed the 100 epochs into high quality so that maximum accuracy can be obtained.

For code refer to Appendix[G] of page.55.

Chapter 6:

RESULTS

6.1. Results of Training the Network

- Epoch 1 finished out of 5 loss : 20867480832.0 success_rate: 1.0
- Epoch 1 finished out of 5 loss : 36023595008.0 success_rate: 1.0
- Epoch 1 finished out of 5 loss : 51028041728.0 success_rate: 1.0
- Epoch 1 finished out of 5 loss : 23555268608.0 success_rate: 1.0
- Epoch 1 finished out of 5 loss : 26307223808.0 success_rate: 1.0
- Accuracy : 0.70

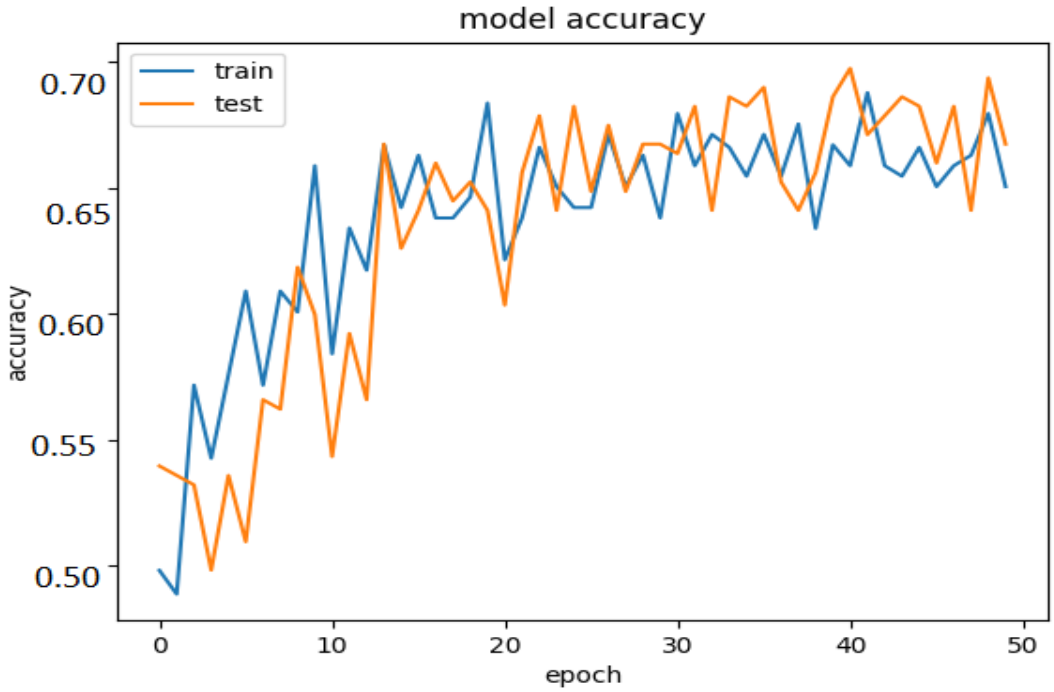

Fig 6.1: Graph of accuracy against number of epoches

6.2. Confusion Matrix

6.2.1.Introduction

If one wants to find out to what extent the Neural Network has performed, or any deep learning network for that matter, after feeding in the data to the model to which the ground truth is already know, then Confusion Matrix is the to go solution, as it gives a very accurate measurement as to how the model has performed

An important aspect of the confusion matrix is that the numerous right and wrong predictions along with the indexed values that are given as a synopsis where each class is partitioned.

The whole idea behind the confusion matrix is that it apart from it giving us the errors that are occurred by a particular class it also informs us about the errors that will eventually be made.

.

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

6.2.3. Definition of Terms

1) Positive (P): Observation was positive.
2) Negative (N): Observation wasn't positive.
3) Observation is positive, and was foreseen to be positive.
4) Observation is positive, but was foreseen negative.
5) Observation is negative, and was foreseen to be negative.
6) Observation is negative, however was foreseen positive.

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

The accuracy of a confusion is defined above.

One of the most popular way to find out the performance of your model is with the employment of a confusion matrix.

In the implementation of our project, the ones basically represent the presence of a cancer cell in the lung while the zeros represent no cancer cell in the lung.

Chapter 7:

ERRORS FACED DURING EXECUTION

The dataset we specified most of the outcomes from the implementation overview, but when it comes to the CNN process, there were many challenges faced

1. Size across the Z axis is not the same throughout

When we say size across the z axis we are basically talking about the number of slices in the DICOM image. There were varied number of slices in the DICOM image. Some reports had 15 slices while the other has 25 slices and so on with varying number.

2. Tensor reshape error

Each and every time we tried to feed a Tensor to the Convolution Neural Network model for it to train, we faced one major obstacle as the model was accepting the shape of the Tensor being fed and would throw a Tensor Reshape Error. The way passed this hurdle was by defining the shape of the Tensor that we were feeding the mod

3. TensorFlow error

After successfully feeding the CNN model with the a tensor who's shape was defined by us, we ran into another obstacle where the GPU was exhausting up it's resources very quickly. After a close introspection we realised that it was because TensorFlow was running on the GPU in the background and the entire GPU was not allocated to this process. So every time a tensor was fed for processing, the kernel had to be restarted and the GPU's memory had to be freed up for the next image. The same issue was faced when we tries to feed the machine with a larger tensor.

4. Insufficient data causing over-fitting

We had obtained exactly of 1600 DICOM reports from LUNA(lung nodule analysis) and the Kaggle website. That was the total size of the data set.

Out of the 1600 reports, 1109 scans were labelled to have cancer, 390 labelled to have cancer while the remaining 101 patients labelled to have cancer.

In out model the predictions are not on spot mainly because of insufficient that was available to us. This posed a problem of overfitting of our model and the only way to overcome this problem of over fitting was by using more data.

Chapter 8:

APPENDIX

[A]Code to keep the number of slices in a DICOM image constant

```
##Data directory
dataDirectory='Lung_Cancer/stage1/stage1/'
lungPatients = os.listdir(dataDirectory)
##Read labels csv
Labels=pd.read_csv('path', index_col=0)
##Setting x*y size to 20
Size = 50
##Setting z-dimensions (number of slices to 20)
NoSlices = 20
```

Since the number of slices in DICOM images for the patient's CT scan is not constant, the above code limits the number of slices to 20 for all the DICOM files so that it runs smoothly during the training and testing phase of the project.

[B] Code to store the processed data as a list

```
def chunks(l, n):
    count = 0
    for i in range(0, len(l), n):
        if (count < NoSlices):
            yield l[i:i + n]
            count = count + 1
#taking average
def mean(l):
    return sum(l) / len(l)

def dataProcessing(patient, labels_df, size=50, noslices=20, visualize=False):
    #getting the value of the lable of patient
    label = labels_df.get_value(patient, 'cancer')
    #assingning the path using patient lable
    path = dataDirectory + patient
    #reading dicom files
    slices = [dicom.read_file(path + '/' + s) for s in os.listdir(path)]
    #sorting the files
    slices.sort(key=lambda x: int(x.ImagePositionPatient[2]))
    new_slices = []
    #resizeing the files and converting to numpy array
    slices = [cv2.resize(np.array(each_slice.pixel_array), (size, size)) for each_slice in slices
    #assingning chunks size using total number of slices
    chunk_sizes = math.floor(len(slices) / noslices)

    #appending to the new_slices list
        for slice_chunk in chunks(slices, chunk_sizes):
            slice_chunk = list(map(mean, zip(*slice_chunk)))
            new_slices.append(slice_chunk)
        #contditions for label
        if label == 1:
            label = np.array([0, 1])
        elif label == 0:
            label = np.array([1, 0])
        #returning new_slice list after converted to numpy array and lable of the patient
        return np.array(new_slices), label
```

[B] Code to store the processed data as a list

```
#appending to the new_slices list
    for slice_chunk in chunks(slices, chunk_sizes):
        slice_chunk = list(map(mean, zip(*slice_chunk)))
        new_slices.append(slice_chunk)
    #contditions for label
    if label == 1:
        label = np.array([0, 1])
    elif label == 0:
        label = np.array([1, 0])
    #returning new_slice list after converted to numpy array and lable of the patient
    return np.array(new_slices), label
```

The above code helped us to reduce the size of the data set from around 160GB to just few Megabytes by concerting it into a NumPy array.

The main idea behind this was to make the entire process faster.

[C] Code to plot histogram

```
def resample(image, scan, new_spacing=[1,1,1]):
    # Determine current pixel spacing
    spacing = np.array([scan[0].SliceThickness] + scan[0].PixelSpacing, dtype=np.float
    resize_factor = spacing / new_spacing
    new_real_shape = image.shape * resize_factor
    new_shape = np.round(new_real_shape)
    real_resize_factor = new_shape / image.shape
    new_spacing = spacing / real_resize_factor
    #The array is zoomed using spline interpolation
    image = scipy.ndimage.interpolation.zoom(image, real_resize_factor, mode='nearest
    return image, new_spacing
    pix_resampled, spacing = resample(first_patient_pixels, first_patient, [1,1,1])
    print("Shape before resampling\t", first_patient_pixels.shape)
```

The above code plots a graph of frequency against the Hounsfield units for the CT scan of one Patient.

[D] Code to plot 3D image of the chest

```
def plot_3d(image, threshold=-300):
    # Position the scan upright,
    # so the head of the patient would be at the top facing the camera
    p = image.transpose(2,1,0)
    verts, faces = measure.marching_cubes(p, threshold)
    fig = plt.figure(figsize=(10, 10))
    ax = fig.add_subplot(111, projection='3d')
    # Fancy indexing: `verts[faces]` to generate a collection of triangles
    mesh = Poly3DCollection(verts[faces], alpha=0.70)
    face_color = [0.45, 0.45, 0.75]
    mesh.set_facecolor(face_color)
    ax.add_collection3d(mesh)
    ax.set_xlim(0, p.shape[0])
    ax.set_ylim(0, p.shape[1])
    ax.set_zlim(0, p.shape[2])
    plt.show()
# print rib cage
plot_3d(pix_resampled, 400)
```

The above code runs to collect all the 2D DICOM slices which are located at different geometric locations and combined to form one 3D image of the chest region.

[E] Code to segment the lung region from the chest region and to fill the lungs with air.

```
def segment_lung_mask(image, fill_lung_structures=True):
    # not actually binary, but 1 and 2.
    # 0 is treated as background, which we do not want
    binary_image = np.array(image > -320, dtype=np.int8)+1
    labels = measure.label(binary_image)

    # Pick the pixel in the very corner to determine which label is air.
    #   Improvement: Pick multiple background labels from around the patient
    #   More resistant to "trays" on which the patient lays cutting the air
    #   around the person in half
    background_label = labels[0,0,0]

    #Fill the air around the person
    binary_image[background_label == labels] = 2
        # Method of filling the lung structures (that is superior to something like
        # morphological closing)
        if fill_lung_structures:
            # For every slice we determine the largest solid structure
            for i, axial_slice in enumerate(binary_image):
                axial_slice = axial_slice - 1
                labeling = measure.label(axial_slice)
                l_max = largest_label_volume(labeling, bg=0)
                if l_max is not None: #This slice contains some lung
                    binary_image[i][labeling != l_max] = 1
        binary_image -= 1 #Make the image actual binary
        binary_image = 1-binary_image # Invert it, lungs are now 1

        # Remove other air pockets insided body
        labels = measure.label(binary_image, background=0)
        l_max = largest_label_volume(labels, bg=0)
        if l_max is not None: # There are air pockets
            binary_image[labels != l_max] = 0
        return binary_image
```

The above code segments the lung region from the chest cavity and fills the lun

[F] Code to build a Convolution Neural Network

```
def convolutional_neural_network(x):   #Build and train a convolutional neural network with
TensorFlow.
    #random_normal to get random values in an array.
    #tf.random creates a variable which is a three-dimensional tensor with shape
tf.random_normal()
    #filled with zeros.
    weights = {'W_conv1':tf.Variable(tf.random_normal([5,5,5,1,32])),
               'W_conv2':tf.Variable(tf.random_normal([5,5,5,32,64])),
               'W_fc':tf.Variable(tf.random_normal([54080,1024])),
               'out':tf.Variable(tf.random_normal([1024, n_classes]))}

    biases = {'b_conv1':tf.Variable(tf.random_normal([32])),
              'b_conv2':tf.Variable(tf.random_normal([64])),
              'b_fc':tf.Variable(tf.random_normal([1024])),
              'out':tf.Variable(tf.random_normal([n_classes]))}

    x = tf.reshape(x, shape=[-1, IMG_PXL_SIZE, IMG_PXL_SIZE, HM_SLICES, 1])#reshape
  x to the shape of [-1, IMG_PXL_SIZE, IMG_PXL_SIZE, HM_SLICES, 1]
 conv1 = tf.nn.relu(conv3d(x, weights['W_conv1']) + biases['b_conv1'])#This layer creates a
convolution kernel that is convolved with the layer input to produce a tensor of outputs.

 conv1 = maxpool3d(conv1)#Performs 3D max pooling on the input.
 conv2 = tf.nn.relu(conv3d(conv1, weights['W_conv2']) + biases['b_conv2'])
 conv2 = maxpool3d(conv2)

 fc = tf.reshape(conv2,[-1, 54080])
 #matmul-matrixmultiplication
 fc = tf.nn.relu(tf.matmul(fc, weights['W_fc'])+biases['b_fc'])
 fc = tf.nn.dropout(fc, keep_rate)#It randomly replaces some values with 0, and increases
the remaining values to compensate.

 output = tf.matmul(fc, weights['out'])+biases['out']
 return output
```

The above code implements a CNN network with the help of TensorFlow

[G] Code to train the neural network

```
def train_neural_network(x):
    #laoding teh data to train
    train_data = np.load('traindata-50-50-20.npy')
    test_data = np.load('testdata-50-50-20.npy')
    train = train_data[:10]
    test = train_data[10:12]
    #calling the convolutional_neural_network function
    prediction = convolutional_neural_network(x)
    #Logits is a function which operates on the unscaled output of earlier layers and on a
linear scale to understand the linear units
    #to computing the cross entropy of the result after the softmax function has been applied.
    cost = tf.reduce_mean(
tf.nn.softmax_cross_entropy_with_logits(logits=prediction,labels=y) )#Computes the mean of
elements across dimensions of a tenso
    # optimize the cost
    optimizer = tf.train.AdamOptimizer().minimize(cost)
    hm_epochs = 5
#initing a TensorFlow Graph object in which tensors are processed through operations
with tf.Session() as sess:
  sess.run(tf.initialize_all_variables())
  for epoch in range(hm_epochs):
    epoch_loss = 0
    success_total = 0
    attempt_total = 0
    for data in train:
      attempt_total += 1
      try:
        X = data[0]
        Y = data[1]
        _, c = sess.run([optimizer, cost], feed_dict={x: X, y: Y})
        epoch_loss += c
        success_total += 1
      except Exception as e:
        print('Error occured')
    print('Epoch', epoch+1, 'completed out of',hm_epochs,'loss:',epoch_loss,
'success_rate:', success_total/attempt_total)
```

[G] Code to train the neural network

```
#compariing prediction and y values
correct = tf.equal(tf.argmax(prediction, 1), tf.argmax(y, 1))

#gtting accuracy using binary values using tf.cast()
accuracy = tf.reduce_mean(tf.cast(correct, 'float'))
print('Accuracy:',accuracy.eval({x: [i[0] for i in test], y: [i[1] for i in test]}))

#calling the function to train the neural network
train_neural_network(x)
return binary_image
```

The above code helps in training the neural network.

Chapter 9:

CONCLUSION

By implementing the project titled ‘Segmentation and prediction from CT scan images for prediction of lung cancer’ there was a lot that we learnt otherwise which we would not have the opportunity to do so.

This project was a huge learning curve and there are a lot of outputs for us to takeaway from this project.

During the first phase, we learnt the various pre-processing techniques that we could apply to an image. The one’s which we learnt about are the mean filter, median filter, gaussian and the bilateral filter.

This project presented us an opportunity to work with 3D Images (DICOM files) with the help of a package called the ‘PyDicom’. We learnt how to work with huge volumes of data by employing TensorFlow. Error handling is another major skill that we have developed throughout the course of the project which will help us in handling errors in a much quicker and efficient manner and also prevent them from even happening in the first place.

Since we had the opportunity to apply the above mentioned concepts onto a real life problem, it gave us a better and deeper understanding.

Chapter 10:

FUTURE WORK

The project that have taken is only at the very begging of its development stage and we desire to carry this project ahead mainly because we believe that much greater improvement can be made to this project by employing Microsoft's ResNets.

The following are the advantages of using Microsoft's ResNets

1) Make more valuable predictions
2) Achieve a much higher value of accuracy
3) Provide a higher learning rate
4) Minimizing the losses

We would wish to carry out this project with the Resnets approach and take this project to the next phase, i.e, the application stage and try and make an impact to the medical industry, contributing in the way we can and have an impact on the life of the individual's.

Chapter 11:

REFERENCE

[1]Qiu, Gnoping. "An improved recursive median filtering scheme for image processing." ***IEEE Transactions on Image Processing*** **5.4 (1996): 646-648.**

[2]Alam, Janee, Sabrina Alam, and Alamgir Hossan. "Multi-stage lung cancer detection and prediction using multi-class svm classifie." ***2018 International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2)*****. IEEE, 2018.**

[3]Sun, Shanhui, Christian Bauer, and Reinhard Beichel. "Automated 3-D segmentation of lungs with lung cancer in CT data using a novel robust active shape model approach." ***IEEE transactions on medical imaging*** **31, no. 2 (2011): 449-460.**

[4]Sun, Shanhui, Christian Bauer, and Reinhard Beichel. "Automated 3-D segmentation of lungs with lung cancer in CT data using a novel robust active shape model approach." ***IEEE transactions on medical imaging*** **31, no. 2 (2011): 449-460.**

[5] Kalaivani, S., Pramit Chatterjee, Shikhar Juyal, and Rishi Gupta. "Lung cancer detection using digital image processing and artificial neural networks." In ***2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*****, vol. 2, pp. 100-103. IEEE, 2017.**