Summer Project On

Title

By

Parnika Tewari (2021510066)

Under the guidance of

Internal Supervisor

Prof. Sakina Banu Salmani

Department of Master Of Computer Application
Sardar Patel Institute of Technology
Autonomous Institute Affiliated to Mumbai University

2022-23

CERTIFICATE OF APPROVAL

This is to certify that the following students

Parnika Tewari (2021510066)

Have satisfactorily carried out work on the project entitled

"Lung Cancer Detection Using Machine Learning"

Towards the fulfilment of project, as laid down by Sardar Patel Institute of Technology during year 2022-23.

Project Guide:
Sakina Banu Salmani

PROJECT APPROVAL CERTIFICATE

This is to certify that the following students

Parnika Tewari (20215100666)

Have successfully completed the Project report on

"Lung Cancer Detection Using Machine Learning",

which is found to be satisfactory and is approved

at

SARDAR PATEL INSTITUTE OF TECHNOLOGY,
ANDHERI (W), MUMBAI

INTERNAL EXAMINER EXTERNAL EXAMINER

HEAD OF DEPARTMENT PRINCIPAL

Contents

Abstract i

Objectives i

List Of Figures ii

List Of Tables ii

1 Introduction 1
- 1.1 Problem Definition . . . 1
- 1.2 Objectives and Scope . . . 1
  - 1.2.1 Objectives . . . 1
  - 1.2.2 Scope . . . 1
- 1.3 Existing System . . . 2
- 1.4 Proposed System . . . 2
- 1.5 System Requirements . . . 3

2 Software Requirement Specification (SRS) and Design 4
- 2.1 Purpose . . . 4
- 2.2 Definition . . . 4
- 2.3 Overall Description . . . 4
  - 2.3.1 Product Functions . . . 4
  - 2.3.2 User Characteristics . . . 5

3 Project Analysis and Design 6
- 3.1 Methodologies Adapted . . . 6
- 3.2 Modules . . . 7
  - 3.2.1 Work Breakdown Structure . . . 7
  - 3.2.2 ResNet50 Model . . . 7
  - 3.2.3 Flowchart . . . 8
  - 3.2.4 PERT Chart . . . 9
  - 3.2.5 Gantt Chart . . . 9

4 Project Implementation and Testing 10
- 4.1 Importing Data and related libraries . . . 10
- 4.2 Preprocessing the data . . . 11
- 4.3 Importing ResNet50 model . . . 12
- 4.4 Model creation . . . 12
- 4.5 Fitting the Data . . . 13
- 4.6 Plotting the result . . . 13
- 4.7 Prediction . . . 14
- 4.8 Code 1 . . . 15
- 4.9 Code 2 . . . 15
- 4.10 Code 3 . . . 16

5 Test Cases **17**

6 Limitations **18**

7 Future Enhancements **18**

8 Bibliography **19**
8.1 Web References 19

Abstract

I have applied the waterfall model of Software Engineering as there is no customer involvement during the course of the development of the project and only end product is to be delivered/demonstrated.

It would be helpful in detecting cancer in early stages both for doctor as there might be a chance if the doctor is unable to detect cancer manually, going through numerous X-ray scans because the cancer nodules are difficult to detect in early stages of tumor growth and are almost invisible to naked eye.

This model will help patients by reducing the cost and frequency of the doctor visits as knowing the effectiveness of the cure through self-evaluation using this model which would spare them from getting fooled too.

Objectives

To develop a chest cancer detection model which facilitates

- Doctors to diagnose the existing cancer nodules in the chest in early phases of tumor generation or cell accumulation which helps in timely treatment of cancer.
- Patients to upload their X-ray images and get the result whether the cancer is still present or not (after long treatment) so that they can act accordingly thus by reducing the cost of seeing the doctor on regular basis and refrain themselves from getting fooled.
- Also, a lightweight image classification model which works on any device with very less space in the memory, training many images at once without overhead and providing an accuracy of approximately 96

List of Figures

3.1.1Diagrammatic Representation of Waterfall Model 6
3.2.1Work Breakdown Structure 7
3.2.2Working of Resnet50 Model 7
3.2.3Flowchart of the Model 8
3.2.4PERT Chart 9
3.2.5Gantt Chart 9
4.1.1 Imported data and libraries 10
4.2.1 Data preprocessing 11
4.3.1Resnet50 model 12
4.4.1Model creation 12
4.5.1Fitting the data into the Model 13
4.6.1Accuracy vs epoch 13
4.7.1Cancer image 14

List of Tables

1.5.1 Hardware Requirements 3
1.5.3 Software Requirements on Server Side 3
1.5.3 Software Requirements on Client Side 3
6.1 Test Case 17

1 Introduction

1.1 Problem Definition

Though we have seen enormous medical advancements in curing cancer yet a special type i.e. lung cancer is still the deadliest of all regardless of gender. Owing to such prevalent nature that cancer posses, there might be a chance if the doctor is unable to detect cancer manually going through numerous X-ray scans, approximately 300 per person while cancer is still present in the chest. Cancer nodules are difficult to detect in early stages of tumor growth and are almost invisible to naked eye. Also, the treatment of cancer is very costly and rigorous as the patient has to visit the doctor frequently which is both expensive and inconvenient.

1.2 Objectives and Scope

1.2.1 Objectives

To develop a chest cancer detection model which facilitates

- Doctors to diagnose the existing cancer nodules in the chest in early phases of tumor generation or cell accumulation which helps in timely treatment of cancer.
- Patients to upload their X-ray images and get the result whether the cancer is still present or not (after long treatment) so that they can act accordingly thus by reducing the cost of seeing the doctor on regular basis and refrain themselves from getting fooled.
- Also, a lightweight image classification model which works on any device with very less space in the memory, training many images at once without overhead and providing an accuracy of approximately 96

1.2.2 Scope

Practicing Doctors can use this model for the study of cancer detection and understand the difference between cancerous and non cancerous images.

As this is a lightweight image classification model which uses ResNet50 neural network layer for detection, it works on less space in memory and is platform independent.

This model will facilitate the doctors for accurate and early detection as well as patients for having the knowledge of the current treatment.

1.3 Existing System

Currently, there is no such system which is patient friendly and they can themselves get predictions for the input X-ray image.
Some of the disadvantages of existing system are as follows :

- Time consuming, Manual work
  Doctor needs to go through the X-ray scans manually which is time consuming and leads to further delay in the treatment.
- High chances of error
  In manual observation, there may be chances of wrong detection in early phases which could lead to drastic effects.
- Unawareness among patients
  In cancer treatments, patients are not much aware of their current situation and they solely depend upon the doctor's word.
- Requirement of large memory and high processor.
  A deep learning model acquire chunks of memory and high processor to train the model and it becomes inconvenient to load and run the model every time.

1.4 Proposed System

As this is a Machine learning model which needs to be deployed to an application, its end users are doctor and patient.

Doctor can diagnose the cancer in its early stages with the help of this model and can begin with the treatment as soon as possible.

Patient can upload their X-ray images and get the prediction whether they are suffering from cancer or not, after going through treatments.

Some of the advantages of our system are as follows :

- User Friendly
  This model is based upon ResNet50 using transfer learning which is a powerful algorithm, providing nearly 96 percent accuracy.
- It is a lightweight model, consumes less memory and is platform independent and portable.

1.5 System Requirements

- Hardware Requirements

Table 1.5.1: Hardware Requirements

Processor	Dual Core Processor or Above
RAM	Minimum 4 GB RAM
Storage	Minimum 8 GB Hard Disk Space for smooth run
GPU	GT Force GPU

- Software Requirements on Server Side

Table 1.5.3: Software Requirements on Server Side

Operating System	OS Independent

- Software Requirements on Client Side

Table 1.5.3: Software Requirements on Client Side

Operating System	Linux/Windows OS
Server	Not Required

2 Software Requirement Specification (SRS) and Design

2.1 Purpose

The purpose of this project is to develop a lung cancer detection model which helps the doctor with early diagnosis of cancer and speedy treatment of it as there might be a chance if the doctor is unable to detect cancer manually, going through numerous X-ray scans.

This model will help patients by reducing the cost and frequency of the doctor visits as knowing the effectiveness of the cure through self-evaluation using this model.

2.2 Definition

To build a lung cancer detection model which takes an X-ray image as input and displays whether cancer is present in that image or not.

2.3 Overall Description

2.3.1 Product Functions

The product function includes a jupyter notebook which contains the path to images, model building , image processing and finally cancer prediction based upon given image.

2.3.2 User Characteristics

There can potentially be two users:

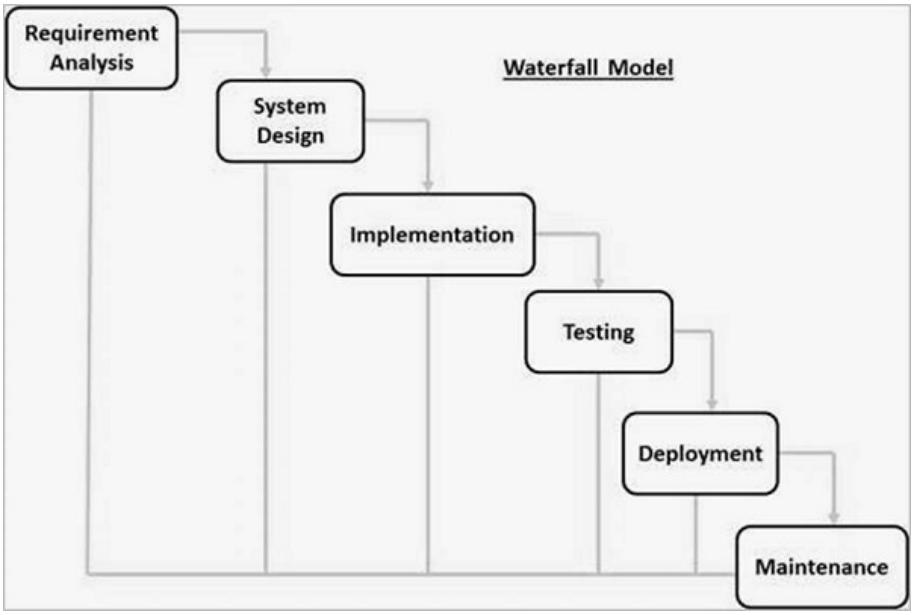
- Doctor: For diagnosing whether the cancer present or not and beginning with early treatment.
- Patient: To know whether he/she is suffering from the cancer after the treatment.

3 Project Analysis and Design

3.1 Methodologies Adapted

I have applied the waterfall model of Software Engineering as there is no customer involvement during the course of the development of the project and only end product is to be delivered/demonstrated.

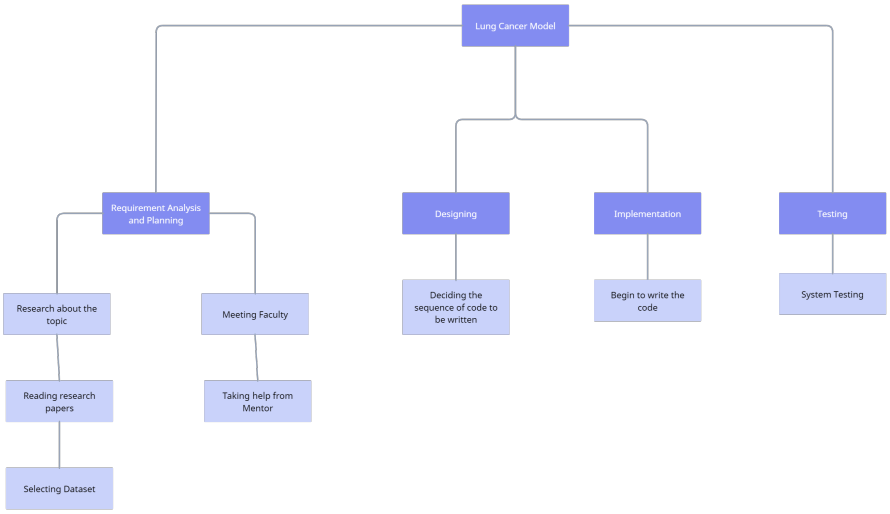
Once the product is developed and if any failure occurs then the cost of such issues is very high, because we need to update everything from document till the logic.



3.1.1: Diagrammatic Representation of Waterfall Model

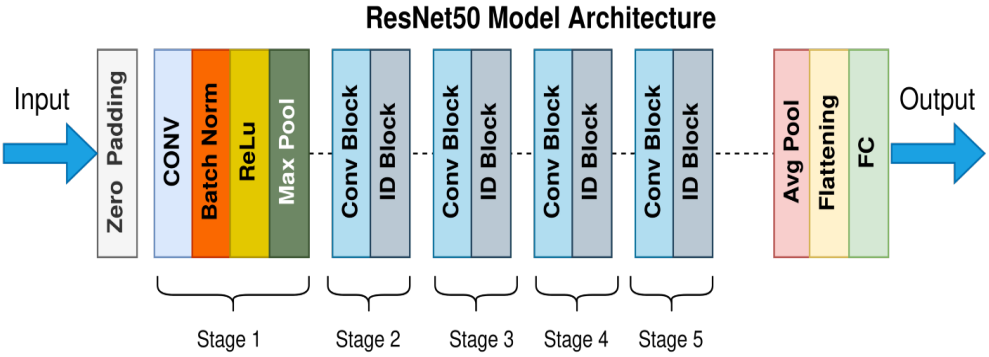
3.2 Modules

3.2.1 Work Breakdown Structure



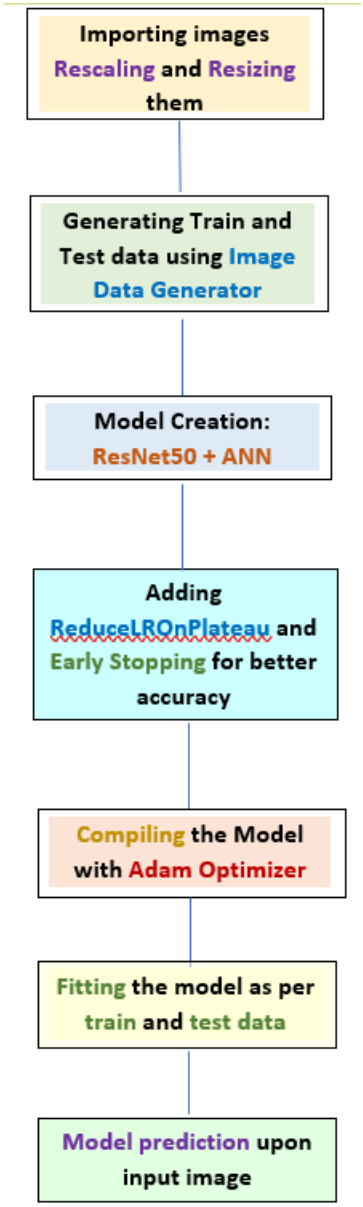
3.2.1: Work Breakdown Structure

3.2.2 ResNet50 Model



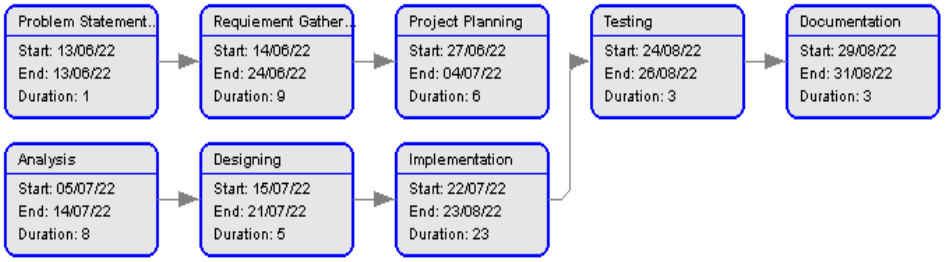
3.2.2: Working of Resnet50 Model

3.2.3 Flowchart



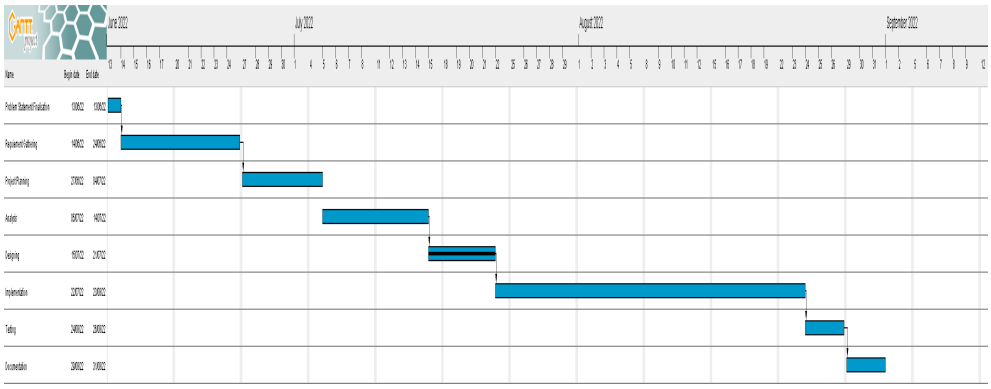
3.2.3: Flowchart of the Model

3.2.4 PERT Chart



3.2.4: PERT Chart

3.2.5 Gantt Chart



3.2.5: Gantt Chart

4 Project Implementation and Testing

4.1 Importing Data and related libraries

In [1]:
```
import tensorflow.keras as keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from tensorflow.keras.models import Sequential
from keras.utils import np_utils
from tensorflow.keras.layers import Dense,Activation, Flatten, Dropout, BatchNormalization, Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras import regularizers, optimizers
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications import resnet, densenet
from tensorflow.keras import Model

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

import PIL
import os
import cv2
```

In [2]:
```
pip install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\dell\anaconda3\lib\site-packages (4.6.0.66)
Requirement already satisfied: numpy>=1.17.3 in c:\users\dell\anaconda3\lib\site-packages (from opencv-python) (1.20.1)
Note: you may need to restart the kernel to use updated packages.
```

In [42]:
```
train_path = '../chest-cancer-dataset/Data/train'
test_path = '../chest-cancer-dataset/Data/test'
valid_path = '../chest-cancer-dataset/Data/valid'

#train folder:
normal_folder = '../normal'
adenocarcinoma_folder = '/adenocarcinoma_left.lower.lobe_T2_N0_M0_Ib'
large_cell_carcinoma_folder = '/large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa'
squamous_cell_carcinoma_folder = '/squamous.cell.carcinoma_left.hilum_T1_N2_M0_IIIa'
```

4.1.1: Imported data and libraries

4.2 Preprocessing the data

In [4]:
```
image_shape = (305,430,3)
N_CLASSES = 4
BATCH_SIZE = 32

train_datagen = ImageDataGenerator(dtype='float32', rescale= 1./255.)
train_generator = train_datagen.flow_from_directory(train_path,
                                                    batch_size = BATCH_SIZE,
                                                    target_size = (305,430),
                                                    class_mode = 'categorical')

valid_datagen = ImageDataGenerator(dtype='float32', rescale= 1./255.)
valid_generator = valid_datagen.flow_from_directory(valid_path,
                                                    batch_size = BATCH_SIZE,
                                                    target_size = (305,430),
                                                    class_mode = 'categorical')

test_datagen = ImageDataGenerator(dtype='float32', rescale = 1.0/255.0)
test_generator = test_datagen.flow_from_directory(test_path,
                                                    batch_size = BATCH_SIZE,
                                                    target_size = (305,430),
                                                    class_mode = 'categorical')
```

```
Found 613 images belonging to 4 classes.
Found 72 images belonging to 4 classes.
Found 315 images belonging to 4 classes.
```

4.2.1: Data preprocessing

4.3 Importing ResNet50 model

```
In [6]: from tensorflow.keras.applications import ResNet50
        from tensorflow.keras import layers

        res_model = ResNet50(include_top=False, pooling='avg', weights='imagenet', input_shape = (350,350, 3))
        res_model.summary()
```

```
Model: "resnet50"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 350, 350, 3       0         []
                             )]

 conv1_pad (ZeroPadding2D)   (None, 356, 356, 3)       0         ['input_1[0][0]']

 conv1_conv (Conv2D)         (None, 175, 175, 64       9472      ['conv1_pad[0][0]']
                             )

 conv1_bn (BatchNormalizati  (None, 175, 175, 64       256       ['conv1_conv[0][0]']
 on)                         )

 conv1_relu (Activation)     (None, 175, 175, 64       0         ['conv1_bn[0][0]']
                             )

 pool1_pad (ZeroPadding2D)   (None, 177, 177, 64       0         ['conv1_relu[0][0]']
```

```
In [7]: for layer in res_model.layers:
            if 'conv5' not in layer.name:
                layer.trainable = False
```

4.3.1: Resnet50 model

4.4 Model creation

```
In [8]: model = Sequential()
        model.add(res_model)
        model.add(Dropout(0.6))
        model.add(layers.Flatten())
        model.add(layers.BatchNormalization())
        model.add(layers.Dense(N_CLASSES, activation='softmax'))
```

```
In [9]: optimizer = optimizers.Adam(learning_rate= 0.00001, decay= 1e-6)
        model.compile(optimizer=optimizer, loss = 'categorical_crossentropy', metrics = ['acc'])
```

```
In [10]: checkpointer = ModelCheckpoint(filepath='../Model/chest_CT_SCAN-ResNet50.hdf5',
                                monitor='val_loss', verbose = 1,
                                save_best_only=True)
         early_stopping = EarlyStopping(verbose=1, patience=15)
```

4.4.1: Model creation

4.5 Fitting the Data

```
In [11]: history_res = model.fit(train_generator,
                                 steps_per_epoch = 20,
                                 epochs = 100,
                                 verbose = 1,
                                 validation_data = valid_generator,
                                 callbacks = [checkpointer, early_stopping])
```

```
Epoch 1/100
20/20 [==============================] - ETA: 0s - loss: 1.3551 - acc: 0.4731
Epoch 1: val_loss improved from inf to 1.41986, saving model to ../Model\chest_CT_SCAN-ResNet50.hdf5
20/20 [==============================] - 530s 26s/step - loss: 1.3551 - acc: 0.4731 - val_loss: 1.4199 - val_acc: 0.2222
Epoch 2/100
20/20 [==============================] - ETA: 0s - loss: 1.0312 - acc: 0.5775
Epoch 2: val_loss improved from 1.41986 to 1.40588, saving model to ../Model\chest_CT_SCAN-ResNet50.hdf5
20/20 [==============================] - 493s 25s/step - loss: 1.0312 - acc: 0.5775 - val_loss: 1.4059 - val_acc: 0.1806
Epoch 3/100
20/20 [==============================] - ETA: 0s - loss: 0.8992 - acc: 0.6248
Epoch 3: val_loss improved from 1.40588 to 1.39596, saving model to ../Model\chest_CT_SCAN-ResNet50.hdf5
20/20 [==============================] - 496s 25s/step - loss: 0.8992 - acc: 0.6248 - val_loss: 1.3960 - val_acc: 0.1806
Epoch 4/100
20/20 [==============================] - ETA: 0s - loss: 0.8065 - acc: 0.6852
Epoch 4: val_loss improved from 1.39596 to 1.39495, saving model to ../Model\chest_CT_SCAN-ResNet50.hdf5
20/20 [==============================] - 491s 25s/step - loss: 0.8065 - acc: 0.6852 - val_loss: 1.3950 - val_acc: 0.1806
Epoch 5/100
20/20 [==============================] - ETA: 0s - loss: 0.7339 - acc: 0.6998
Epoch 5: val_loss did not improve from 1.39495
```

```
In [37]: model.save("../Model/Final_model.h5")
```

4.5.1: Fitting the data into the Model

4.6 Plotting the result

```
In [21]: (history_res.history['accu'][-1])*100
```

```
Out[21]: 96.90048694610596
```

```
In [33]:
def display_training_curves(training, validation, title, subplot):
    if subplot%10==1: # set up the subplots on the first call
        plt.subplots(figsize=(10,10), facecolor='#F0F0F0')
        plt.tight_layout()
    ax = plt.subplot(subplot)
    ax.set_facecolor('#F8F8F8')
    ax.plot(training)
    ax.plot(validation)
    ax.set_title('model '+ title)
    ax.set_ylabel(title)
                            #ax.set_ylim(0.28,1.05)
    ax.set_xlabel('epoch')
    ax.legend(['train', 'valid'])
```

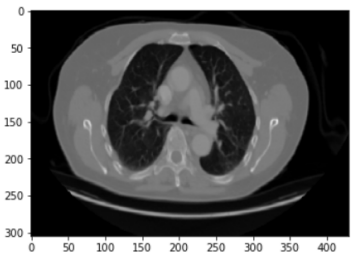
```
In [36]: display_training_curves(history_res.history['acc'], history_res.history['val_acc'], 'accuracy', 212)
```

4.6.1: Accuracy vs epoch

4.7 Prediction

```
In [83]: path = "../chest-cancer-dataset/Data/test/adenocarcinoma/000114.png"
         chestScanPrediction(path,model)
```

```
1/1 [==============================] - 0s 381ms/step
```




```
Adenocarcinoma
```

4.7.1: Cancer image

4.8 Code 1

In [11]:
```
history_res = model.fit(train_generator,
                        steps_per_epoch = 20,
                        epochs = 100,
                        verbose = 1,
                        validation_data = valid_generator,
                        callbacks = [checkpointer, early_stopping])
```

```
Epoch 1/100
20/20 [==============================] - ETA: 0s - loss: 1.3551 - acc: 0.4731
Epoch 1: val_loss improved from inf to 1.41986, saving model to ../Model\chest_CT_SCAN-ResNet50.hdf5
20/20 [==============================] - 530s 26s/step - loss: 1.3551 - acc: 0.4731 - val_loss: 1.4199 - val_acc: 0.2222
Epoch 2/100
20/20 [==============================] - ETA: 0s - loss: 1.0312 - acc: 0.5775
Epoch 2: val_loss improved from 1.41986 to 1.40588, saving model to ../Model\chest_CT_SCAN-ResNet50.hdf5
20/20 [==============================] - 493s 25s/step - loss: 1.0312 - acc: 0.5775 - val_loss: 1.4059 - val_acc: 0.1806
Epoch 3/100
20/20 [==============================] - ETA: 0s - loss: 0.8992 - acc: 0.6248
Epoch 3: val_loss improved from 1.40588 to 1.39596, saving model to ../Model\chest_CT_SCAN-ResNet50.hdf5
20/20 [==============================] - 496s 25s/step - loss: 0.8992 - acc: 0.6248 - val_loss: 1.3960 - val_acc: 0.1806
Epoch 4/100
20/20 [==============================] - ETA: 0s - loss: 0.8065 - acc: 0.6852
Epoch 4: val_loss improved from 1.39596 to 1.39495, saving model to ../Model\chest_CT_SCAN-ResNet50.hdf5
20/20 [==============================] - 491s 25s/step - loss: 0.8065 - acc: 0.6852 - val_loss: 1.3950 - val_acc: 0.1806
Epoch 5/100
20/20 [==============================] - ETA: 0s - loss: 0.7339 - acc: 0.6998
Epoch 5: val_loss did not improve from 1.39495
```

In [37]:
```
model.save("../Model/Final_model.h5")
```

4.9 Code 2

In [1]:
```
import tensorflow.keras as keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from tensorflow.keras.models import Sequential
from keras.utils import np_utils
from tensorflow.keras.layers import Dense,Activation, Flatten, Dropout, BatchNormalization, Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras import regularizers, optimizers
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications import resnet, densenet
from tensorflow.keras import Model

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

import PIL
import os
import cv2
```

In [2]:
```
pip install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\dell\anaconda3\lib\site-packages (4.6.0.66)
Requirement already satisfied: numpy>=1.17.3 in c:\users\dell\anaconda3\lib\site-packages (from opencv-python) (1.20.1)
Note: you may need to restart the kernel to use updated packages.
```

In [42]:
```
train_path = '../chest-cancer-dataset/Data/train'
test_path = '../chest-cancer-dataset/Data/test'
valid_path = '../chest-cancer-dataset/Data/valid'

#train folder:
normal_folder = '../normal'
adenocarcinoma_folder = '/adenocarcinoma_left.lower.lobe_T2_N0_M0_Ib'
large_cell_carcinoma_folder = '/large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa'
squamous_cell_carcinoma_folder = '/squamous.cell.carcinoma_left.hilum_T1_N2_M0_IIIa'
```

4.10 Code 3

In [4]:
```
image_shape = (305,430,3)
N_CLASSES = 4
BATCH_SIZE = 32

train_datagen = ImageDataGenerator(dtype='float32', rescale= 1./255.)
train_generator = train_datagen.flow_from_directory(train_path,
                                                    batch_size = BATCH_SIZE,
                                                    target_size = (305,430),
                                                    class_mode = 'categorical')

valid_datagen = ImageDataGenerator(dtype='float32', rescale= 1./255.)
valid_generator = valid_datagen.flow_from_directory(valid_path,
                                                    batch_size = BATCH_SIZE,
                                                    target_size = (305,430),
                                                    class_mode = 'categorical')

test_datagen = ImageDataGenerator(dtype='float32', rescale = 1.0/255.0)
test_generator = test_datagen.flow_from_directory(test_path,
                                                    batch_size = BATCH_SIZE,
                                                    target_size = (305,430),
                                                    class_mode = 'categorical')
```

```
Found 613 images belonging to 4 classes.
Found 72 images belonging to 4 classes.
Found 315 images belonging to 4 classes.
```

5 Test Cases

Table 6.1: Test Case

Test Case ID	Test Case Name	Test Data	Expected Output	Actual Output	Result
1	Image is cancerous or not	Path to Adenocarcinoma Image	Adenocarcinoma	Adenocarcinoma	Pass
2	Image is cancerous or not	Path to Large Cell Carcinoma Image	Large Cell Carcinoma	Large Cell Carcinoma	Pass
3	Image is cancerous or not	Path to Squamous Cell Carcinoma Image	Squamous Cell Carcinoma	Squamous Cell Carcinoma	Pass
4	Image is cancerous or not	Path to Normal Image	Normal	Normal	Fail

6 Limitations

- Accuracy of the proposed system is 96 percent which can be further improved to have better predictions.
- Model takes a lot of time in training the data which is approximately 10 hours on an average.
- Fitting the model against validation data happens abruptly.
- The model does not specify the region of the cancer nodules if present.

7 Future Enhancements

- Increasing the testing accuracy.
- Reducing the time period of training the model.
- Fitting the model against the validation data, smoothly.
- Identifying the area of cancer nodules.

8 Bibliography

8.1 Web References

[1.] https://www.kaggle.com/datasets/mohamedhanyyy/chest-ctscan-images
[2.] https://paperswithcode.com/paper/knowledge-based-analysis-for-mortality
[3.] https://www.researchgate.net/publication/324728060_Lung_Cancer_
\protect\@normalcr\relaxDetection_using_Deep_Learning
[4.] https://keras.io/guides/transfer_learning/#:~:text=Setting%20layer.
trainable%20to%20False%20moves%20all%20the%20layer%27s,that%20relies%
20on%20trainable_weights%20to%20apply%20gradient%20updates%29
[5.] https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33
[6.] https://keras.io/api/applications/resnet/