A Project Report

on

LUNG CANCER DETECTION USING SCAN IMAGES

Submitted in partial fulfillment of the

requirement for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

Under The Supervision of
Mrs. Aanchal Vij
Assistant Professor
Department of Computer Science and Engineering

Submitted By

Kartik Kumar – 19SCSE1010673
Shivank Srivastava – 19SCSE1180048

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING,
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
GALGOTIAS UNIVERSITY, GREATER NOIDA,
INDIA DECEMBER, 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **"LUNG CANCER DETECTION USING SCAN IMAGES"** in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the **School of Computing Science and Engineering of Galgotias University**, Greater Noida, is an original work carried out during the period of **JULY-2021 to DECEMBER 2021**, under the supervision of **Mrs. Aanchal Vij, Assistant Professor, Department of Computer Science and Engineering** of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Kartik Kumar (19SCSE1010673)

Shivank Srivastava (19SCSE1180048)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor Name

(Mrs. Aanchal Vij, Assistant Professor)

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **Kartik Kumar – 19SCSE1010673 Shivank Srivastava – 19SCSE1180048** has been held on _____ and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING.**

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date:

Place:

ACKNOWLEDGMENT

We would like to show our thanks to Ms. Aanchal Vij, Project Mentor, Galgotias University, who introduced us to the methodology of work and whose enthusiasm for the "underlying structures" had a lasting impact during multiple consultations and for providing us with good guidance for assignment. In writing this article, we would also like to extend our sincere gratitude to all those who have directly and indirectly guided us.

ABSTRACT

Lung cancer is one of the most dangerous and deadly diseases in the world. However, early diagnosis and treatment can save lives. Although, CT scan scanning is the best way to take pictures in the medical field, it is difficult for doctors to interpret and diagnose cancer on CT scans. Therefore, computer-assisted diagnostics may be helpful for doctors to diagnose cancer cells accurately. Many computer assistants using imaging and machine learning techniques have been researched and used. The main purpose of this study was to explore the various computer-assisted techniques, analyses the best current method and identify their limitations and their constraints and finally propose a new model with progress to the current best model. The method used was for lung cancer screening techniques to be sorted and listed on the basis of their accurate diagnosis. Strategies are analyzed at each step and the overall scope, barriers are identified. It is found that some have low accuracy and some have high accuracy but are not close to 100%. Therefore, our research aims to increase accuracy to 100%.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER-01

Introduction

1.1 Introduction

One of the biggest causes of cancer-related death is lung cancer. It is difficult to see because it appears and shows signs of final phase. However, the mortality rate and chances are reduced by early detection and treatment of the disease. The best CT imaging method is reliable in diagnosing lung cancer because it can expose all suspects and undiagnosed lung cancer tumors [1]. However, the variability in intensity in CT scan and anatomical images the structure of adverse judgment by physicians and radiologists may cause difficulties in marking a cancer cell [2]. Recently, to help radiologists and physicians accurately diagnose cancer Computer Aid Diagnosis has evolved an additional and promising tool [3]. There have been many programs developed and ongoing research on lung detection cancer. However, some systems do not have the exact accuracy of detection and some systems still have to has been upgraded to achieve 100% maximum accuracy. Graphics processing techniques and machine learning techniques for diagnosing and classifying lung cancer have been used. We've read the latest programs designed cancer screening based on CT scans of the lungs to select the latest best systems and diagnostics to them and a new model was propose.

1.2 Formulation of Problem

Lung cancer is one of the most dangerous and deadly diseases in the world. However, early diagnosis and treatment can save lives. Although, CT scan scanning is the best way to take pictures in the medical field, it is difficult for doctors to interpret and diagnose cancer on CT scans. Therefore, computer-assisted diagnostics may be helpful for doctors to diagnose cancer cells accurately. Many computer assistants using imaging and machine learning techniques have been researched and used.

1.2.1 Tools and technology used

i. Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

Keras is:

- **Flexible** -- Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.

- **Powerful** -- Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, or Waymo.

Figure 1: Keras

ii. TensorFlow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

Figure 2: TensorFlow

iii. Kaggle

Kaggle, a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

Figure 3: Kaggle

iv. Pandas

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Figure 4: Pandas

v. NumPy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

Figure 5: NumPy

vi. Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats .
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

matplotlib.pyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.

Figure 6: Matplotlib

CHAPTER-02

Literature Survey/Project Design

Several researchers have proposed and implemented lung cancer detection using various image processing and machine learning methodologies. Agarwal, Furquan, and Kalra [4] proposed a model for distinguishing nodules from normal lung anatomical structure. Geometrical, statistical, and Gray level properties are extracted using this method. For segmentation, LDA is utilised as a classifier and optimal thresholding is applied. The method has an accuracy of 84 percent, a sensitivity of 97.14 percent, and a specificity of 53.33 percent. Despite the fact that the technology detects the cancer nodule, its accuracy is still poor. To classify, no machine learning approaches were applied; instead, standard segmentation techniques were used. As a result, combining any of its processes in our new model does not increase the likelihood of improvement.

In his CAD system, Jin, Zhang, and Jin [5] used a convolution neural network as a classifier to identify lung cancer. The method has an accuracy of 84.6 percent, a sensitivity of 82.5 percent, and a specificity of 86.7 percent. This model has the advantage of using a circular filter in the Region of interest (ROI) extraction phase, which minimises the cost of training and recognition steps. Although the implementation cost is lowered, the accuracy remains inadequate. For grouping or segmentation, Sangamithraa and Govindaraju [6] employ the K mean unsupervised learning approach. It categorises the pixel dataset based on specified properties. This

model employs a back propagation network for categorization. The gray-level cooccurrence grid (GLCG) approach is used to extract features such as entropy, correlation, homogeneity, PSNR, and SSIM. The system has a 90.7 percent accuracy. For noise reduction, an image pre-processing median filter is utilised, which can be effective for our new model to eliminate noise and enhance accuracy.

Roy, Sirohi, and Patle [7] used a fuzzy interference system and an active contour model to construct a system for detecting lung cancer nodules. For visual contrast enhancement, this method employs grey transformation. Before segmentation, the image is binarized, then the resulting image is segmented using an active contour model. The fuzzy inference method is used to classify cancer. To train the classifier, features such as area, mean, entropy, correlation, main axis length, and minor axis length are extracted. The system's overall accuracy is 94.12 percent. Among its limitations, it does not identify cancer as benign or malignant, which is the suggested model's future scope.

Ignatious and Joseph [8] created a mechanism that makes use of watershed segmentation. The Gabor filter is used in pre-processing to improve image quality. It contrasts the accuracy of the neural fuzzy model and the region growing method. The proposed model's accuracy is 90.1 percent, which is greater than the model with segmentation using a neural fuzzy model and the region increasing approach. The advantage of this model is that it employs marker-controlled watershed segmentation, which eliminates the problem of over segmentation. As a shortcoming,

it does not distinguish between benign and malignant cancers, and while accuracy is good, it is still not sufficient. Some adjustments and contributions to this model have the potential to increase the accuracy to a reasonable level.

Gonzalez and Ponomarev [9] proposed a system for determining whether lung cancer is benign or malignant. The priori information and House field Unit (HU) are used by the system to determine the Region of Interest (ROI). Shape features such as area, eccentricity, circularity, and fractal dimension are retrieved, as are textural features such as mean, variance, energy, entropy, skewness, contrast, and smoothness, to train and classify the support vector machine to determine if the nodule is benign or cancerous. This model has the advantage of classifying cancer as benign or malignant; nevertheless, it has the limitation of requiring prior information about the region of interest. The classification of benign or malignant tumours by the model utilizing support vector machine may be useful in our new model.

Based on an analysis of relevant literature, the approach proposed by Ignatius and Joseph [8] is the current best option based on accuracy and the benefits of the stages performed. In image pre-processing, the Gabor filter is used to enhance the image, and the marker-controlled watershed approach is used to segment the image and find the cancer nodule. This model also extracts simply the cancer nodule's area, perimeter, and eccentricity. It compares itself to other previously presented models and highlights its accuracy of 90.1 percent, which is higher than others.

Even though the system is currently the greatest answer (see Figure 1), it has certain limitations. They are indicated in red below :

- For cancer nodules, just a few traits have been retrieved.
- No pre - processing, such as removal of noise or image smoothing, has been applied, which may aid in the accurate detection of nodules.
- No classification of removed cancer as malignant or benign has been conducted.

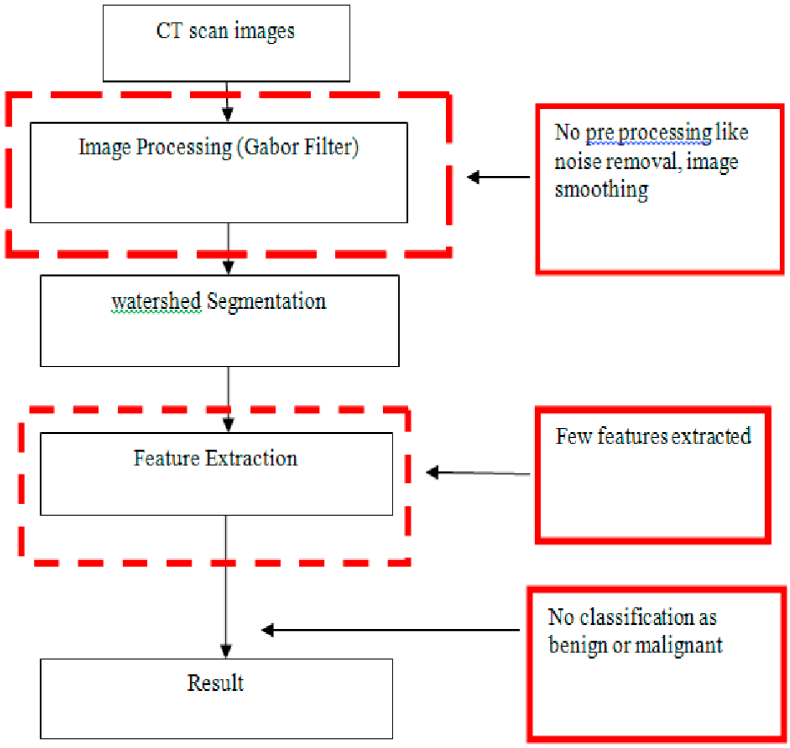

Figure 7: Processing

CHAPTER-03

Functionality/Working of Project

3.1 Working:

Changes have been made to the existing best solution, and a new model is proposed in Figure Instead of the basic machine learning algorithm like support vector machine, decision tree, and random forest, the simple CNN, VGG16, InceptionV3 and ResNet50 were used for improving the accuracy of the model in simple CNN we have used four convolutional layer with input shape of (350,350,3) and got our base model accuracy and after that we used VGG16 model with the same input shape and got the higher accuracy then simple CNN model that is 72.38% then we used InceptionV3 model and got the higher accuracy then VGG16 model that is 78.73 % then we also used ResNet50 Model and we got out highest accuracy which is 79.04%. The best model concludes with the detection of a lung cancer types i.e., Adenocarcinoma, large cell carcinoma and squamous cell carcinoma. As a result, a lung cancer detection was done using RestNet50 Model.

Convolutional Neural Network

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data.

CNNs are powerful image processing, artificial intelligence (AI) that use deep learning to perform both generative and descriptive tasks, often using machine vison that includes image and video recognition, along with recommender systems and natural language processing (NLP).

To begin, we have used a simple CNN with input shape of (350,350,3) with four layers of convolutional and three layers of max pooling. In Convolutional layer we have used 32 filters with kernel size of (3,3) and “RELU” activation function after four convolution and three pooling layer we used dropout with rate of 0.25 then Dense layer with “RELU” activation function then one more Dense layer with rate of 0.5 and then again Dense Layer with “Sigmoid” activation function and then uses a “Adam” optimizer. As a result, we got the base model accuracy which is 47.30%. Below figure will show the graph of model accuracy and model loss with respect to the epochs.

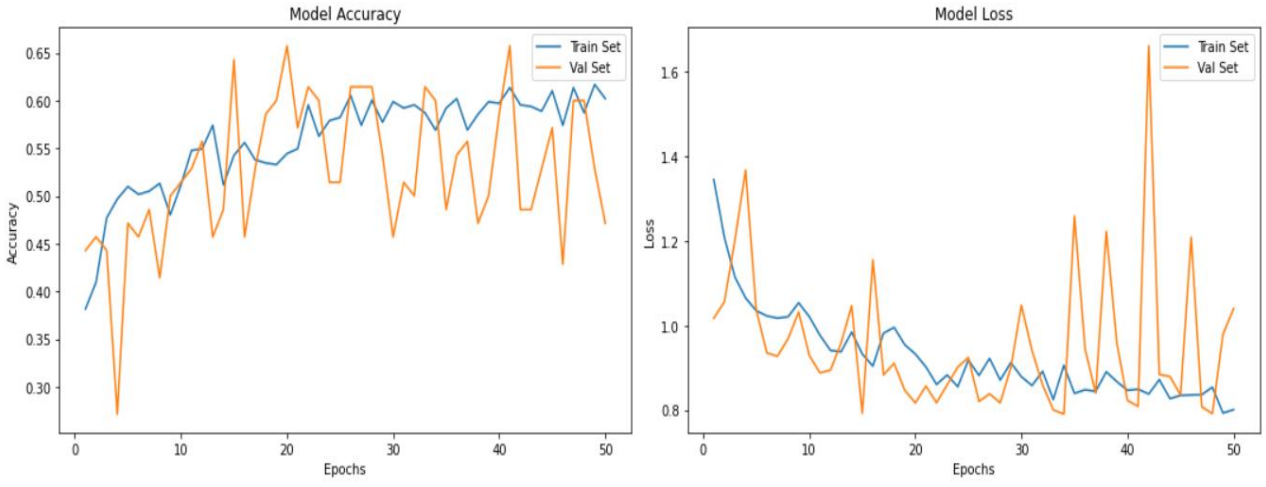

Figure 8: CNN Model Outcome

VGG16 Model

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual computer vision competition. Each year, teams compete on two tasks. The first is to detect objects within an image coming from 200 classes, which is called object localization. The second is to classify images, each labeled with one of 1000 categories, which is called image classification. VGG 16 was proposed by Karen Simonyan and Andrew Zisserman of the Visual Geometry Group Lab of Oxford University in 2014 in the paper "VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION".

In VGG16 model we have used the base model of VGG16 with two Dense layers in which one dense layer use of "RELU" activation function and last one uses "Sigmoid" activation function and one Dropout layer with rate of 0.25. As a result, we got the better accuracy then the simple CNN that is 72.38%. In below figure will show the graph of model accuracy and model loss with respect to the epochs.

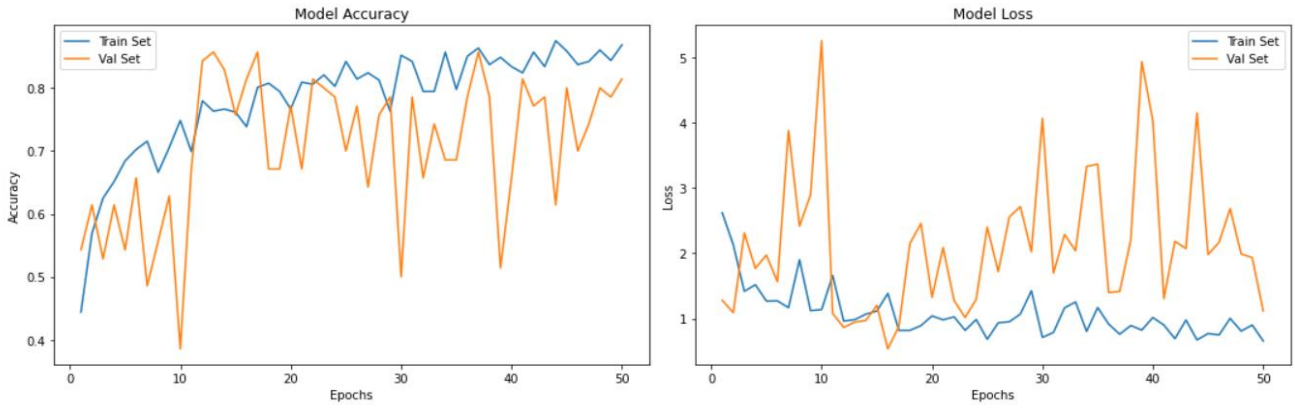

Figure 9: VGG16 Model Outcome

InceptionV3 Model

Inception v3 is a widely-used image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The model is the culmination of many ideas developed by multiple researchers over the years. In InceptionV3 model we have used the base model of InceptionV3 with two Dense layers in which one dense layer use of “RELU” activation function and last one uses “Sigmoid” activation function and one Dropout layer with rate of 0.20. As a result, we got the better accuracy then the simple VGG16 that is 78.73%. In below figure will show the graph of model accuracy and model loss with respect to the epochs.

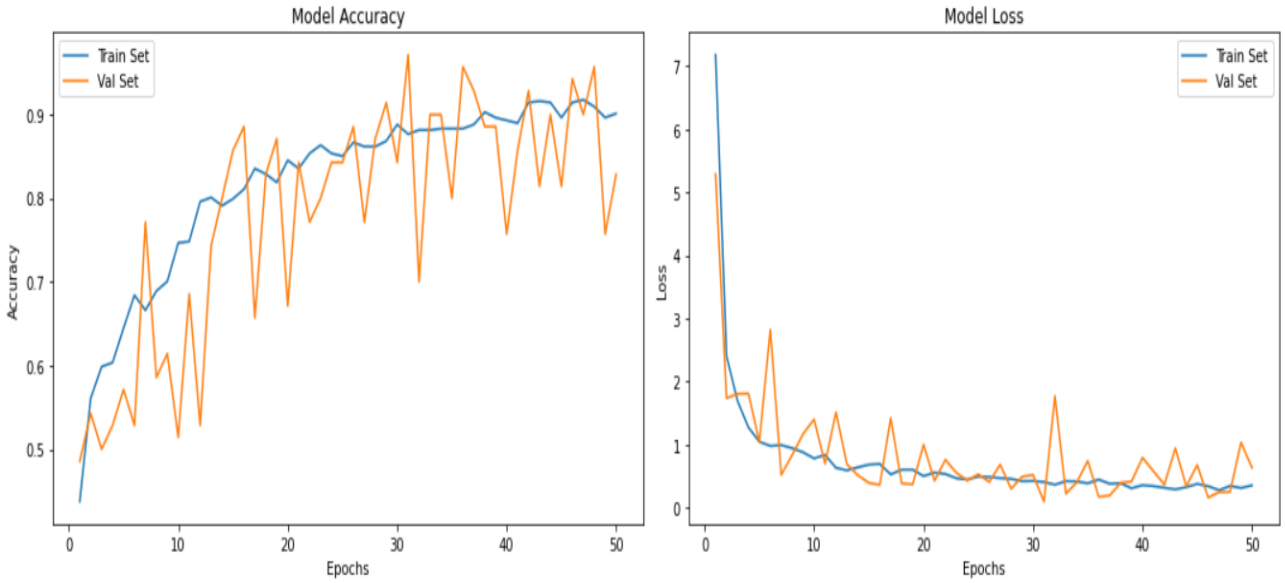

Figure 10: Inception V3 Model Outcome

ResNet Model

ResNet, short for Residual Network is a specific type of neural network that was introduced in 2015 by Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun in their paper “Deep Residual Learning for Image Recognition”.The ResNet models were extremely successful which you can guess from the following:

- Replacing VGG-16 layers in Faster R-CNN with ResNet-101. They observed relative improvements of 28%
- Efficiently trained networks with 100 layers and 1000 layers also.

In ResNet model we have used the base model of ResNet with only One Dense layers in which we used “Sigmoid” activation function and No Dropout layer. As a result, we got the better accuracy then the simple InceptionV3 that is 79.04%. In below figure will show the graph of model accuracy and model loss with respect to the epochs.

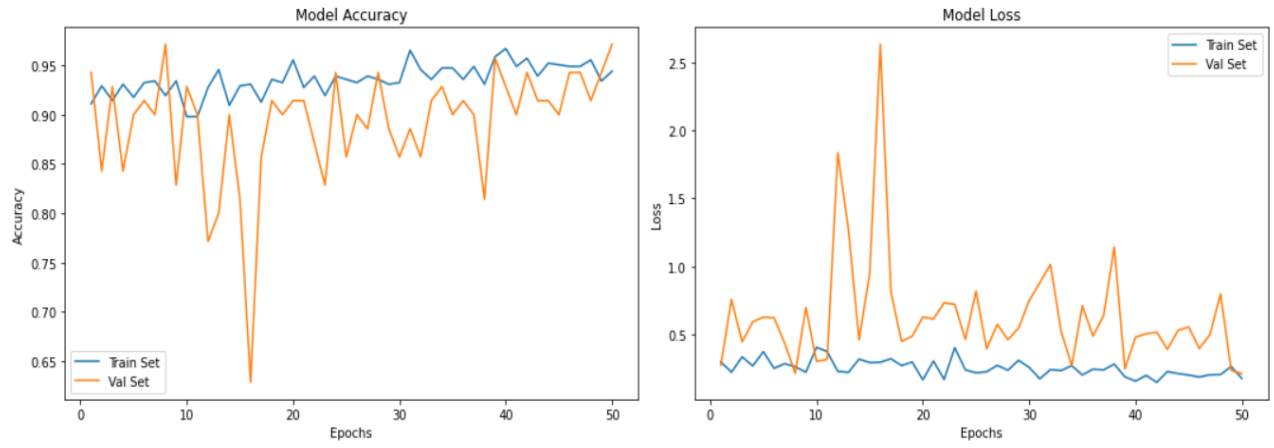

Figure 11: ResNet Model Outcome

Classification

This step determines whether type of cancer is Adenocarcinoma, large cell carcinoma and squamous cell carcinoma Like a classifier, the ResNet is being used. It is an Architecture of deep learning. ResNet50 is a variant of ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. It has 3.8 x 10^9 Floating points operations. It is a widely used ResNet model and we have explored ResNet50 architecture in depth [1].

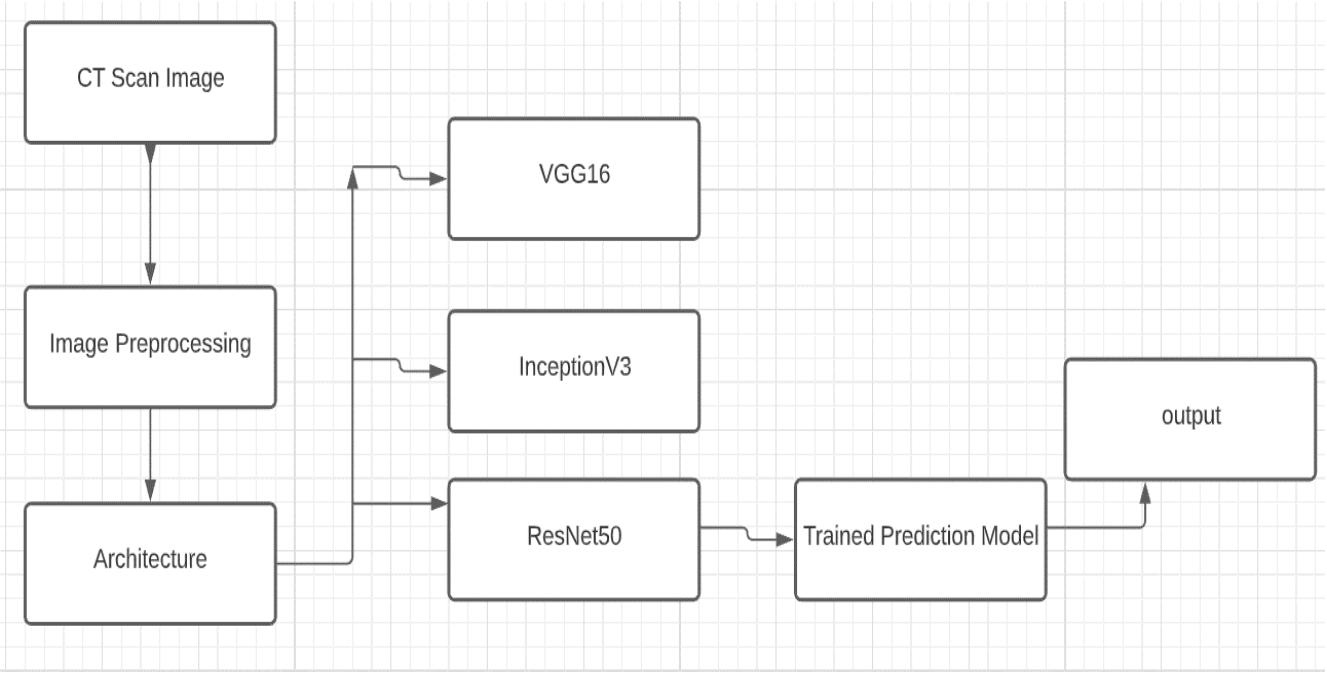

Figure 12: Diagrammatic Representation of Processing our model

Comparison

In below figure, we have shown the comparison between different-different architecture that we used in our model.

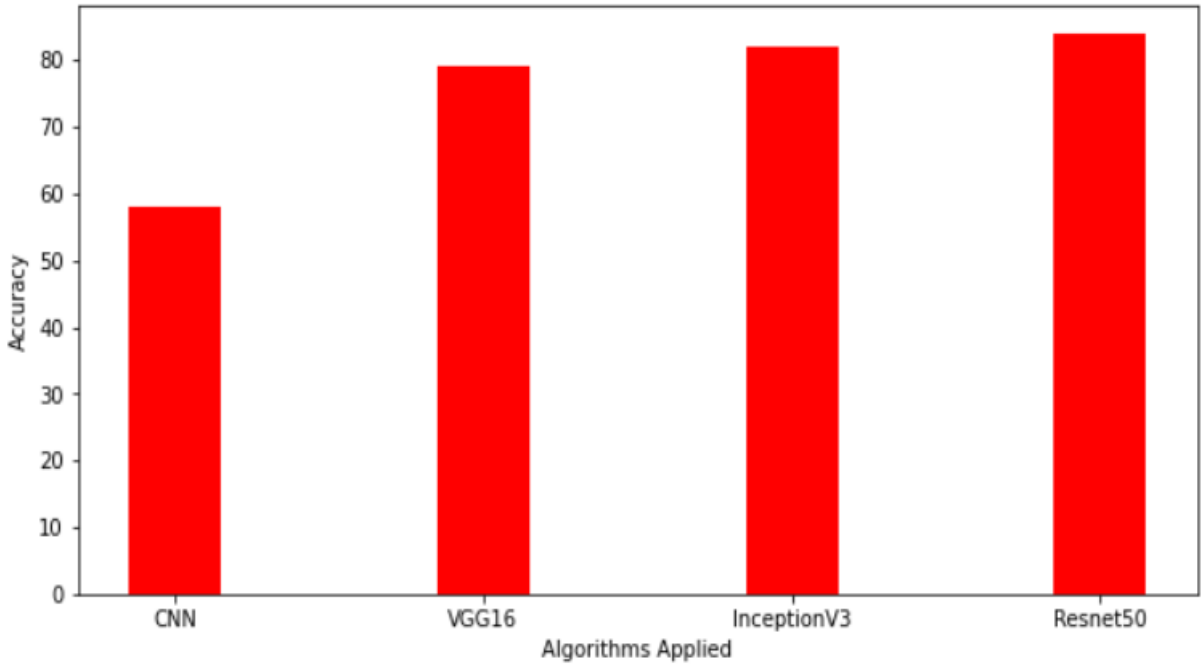

Figure 13: Comparison between Outputs

3.2 STRENGTHS AND WEAKNESS

The following are the Strength of the Model:

- Experiments A far better contemporary model exists for increasing the accuracy of detection of lung cancer.
- It categorizes lung cancer that has been determined to be serious or hazardous.
- Attempts to eliminate the salt-pepper and speckle noises that generate noncancerous diagnoses.

Along with its strengths, the model has several flaws. They are listed below:

- There is an improvement in accuracy, but it is still not at the best level, which is close to 100 percent.
- It categorizes cancer as Adenocarcinoma, large cell carcinoma and squamous cell carcinoma.

3.3 CHALLENGES AND PROBLEM

- **GPU**

For training model, we need more amount of GPU and we need faster GPU that train the model very fast. Training models is a hardware intensive task, and a decent GPU will make sure the computation of neural networks goes smoothly.

- **Training time**

Model takes a lot of time to train even on multiple GPUs. For training every model and checking different parameters take lots of time.

- **Quality of Service**

As a large amount of data for various services is migrated, there may be a lack of service consistency. Steps must be done to guarantee that the standard measure provides improved facilities for a wide range of applications around the world.

- **Security Attacks**

Knowledge derived from devices and objects in this smart world is vulnerable to security threats such as first-hand attacks, rumor attacks, analysis assaults, assumption attacks, and automated invasion attacks. A proper protective structure should be developed to counter these dangers.

CHAPTER-04

Result and Discussions

4.1 SOURCE CODE:

```
## Importing Libraries

import os

import cv2

import pandas as pd

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings("ignore")

# File Directory for both the train and test

train_path = "../input/chest-ctscan-images/Data/train"

val_path = "../input/chest-ctscan-images/Data/valid"

test_path = "../input/chest-ctscan-images/Data/test"

def GetDatasetSize(path):
```

```
    num_of_image = {}
    for folder in os.listdir(path):
        # Counting the Number of Files in the Folder
        num_of_image[folder] = len(os.listdir(os.path.join(path, folder)));
    return num_of_image;
train_set = GetDatasetSize(train_path)
val_set = GetDatasetSize(val_path)
test_set = GetDatasetSize(test_path)
print(train_set,"\n\n",val_set,"\n\n",test_set)
labels = ['squamous.cell.carcinoma', 'normal', 'adenocarcinoma', 'large.cell.carcinoma']
train_list = list(train_set.values())
val_list = list(val_set.values())
test_list = list(test_set.values())
x = np.arange(len(labels))  # the label locations
width = 0.25  # the width of the bars
fig, ax = plt.subplots()
rects1 = ax.bar(x - width, train_list, width, label='Train')
```

```
rects2 = ax.bar(x, val_list, width, label='Val')

rects3 = ax.bar(x + width, test_list, width, label='Test')

# Add some text for labels, title and custom x-axis tick labels, etc.

ax.set_ylabel('Count of Images')

ax.set_title('Lung Cancer Dataset')

ax.set_xticks(x, labels)

plt.xticks(rotation=15)

ax.legend()

ax.bar_label(rects1)

ax.bar_label(rects2)

ax.bar_label(rects3)

fig.tight_layout()

plt.show()

## Importing Keras for Image Classification

import tensorflow.keras

from tensorflow.keras import layers

from tensorflow.keras import Model
```

```
from tensorflow.keras.models import Sequential

from tensorflow.keras.preprocessing import image

from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input

from tensorflow.keras.models import load_model, Model

from tensorflow.keras.layers import Dense,Conv2D, Flatten, MaxPool2D, Dropout

train_datagen = ImageDataGenerator(rescale = 1.0/255.0,

                                   horizontal_flip = True,

                                   fill_mode = 'nearest',

                                   zoom_range=0.2,

                                   shear_range = 0.2,

                                   width_shift_range=0.2,

                                   height_shift_range=0.2,

                                   rotation_range=0.4)


train_data = train_datagen.flow_from_directory(train_path,
```

```
                                        batch_size = 5,

                                        target_size = (350,350),

                                        class_mode = 'categorical')

train_data.class_indices

val_datagen = ImageDataGenerator(rescale = 1.0/255.0)

val_data = val_datagen.flow_from_directory(val_path,

                                        batch_size = 5,

                                        target_size = (350,350),

                                        class_mode = 'categorical')

val_data.class_indices

test_datagen = ImageDataGenerator(rescale = 1.0/255.0)

test_data = test_datagen.flow_from_directory(test_path,

                                        batch_size = 5,

                                        target_size = (350,350),

                                        class_mode = 'categorical')

test_data.class_indices

## CNN Model
```

```
model = Sequential()

# Convolutional Layer with input shape (350,350,3)

model.add(Conv2D(filters=32, kernel_size= (3,3), activation= 'relu', input_shape=(350,350,3)) )

model.add(Conv2D(filters=32, kernel_size=(3,3), activation='relu' ))

model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu' ))

model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(filters=128, kernel_size=(3,3), activation='relu' ))

model.add(MaxPool2D(pool_size=(2,2)))

model.add(Dropout(rate=0.25))

model.add(Flatten())

model.add(Dense(units=64, activation='relu'))

model.add(Dropout(rate=0.5))

model.add(Dense(units=4, activation='sigmoid'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']  )

model.summary()

# Adding Model check point Callback
```

```
mc = ModelCheckpoint(
    filepath="./ct_cnn_best_model.hdf5",
    monitor= 'val_accuracy',
    verbose= 1,
    save_best_only= True,
    mode = 'auto'
    );
call_back = [mc];
# Fitting the Model
cnn = model.fit(
    train_data,
    steps_per_epoch = train_data.samples//train_data.batch_size,
    epochs = 50,
    validation_data = val_data,
    validation_steps = val_data.samples//val_data.batch_size,
    callbacks = call_back
    )
```

```
# Loading the Best Fit Model

model = load_model("./ct_cnn_best_model.hdf5")

# Checking the Accuracy of the Model

accuracy_cnn = model.evaluate_generator(generator= test_data)[1]

print(f"The accuracy of the model is = {accuracy_cnn*100} %")

cnn.history.keys()

# Plot model performance

acc = cnn.history['accuracy']

val_acc = cnn.history['val_accuracy']

loss = cnn.history['loss']

val_loss = cnn.history['val_loss']

epochs_range = range(1, len(cnn.epoch) + 1)

plt.figure(figsize=(15,5))

plt.subplot(1, 2, 1)

plt.plot(epochs_range, acc, label='Train Set')

plt.plot(epochs_range, val_acc, label='Val Set')

plt.legend(loc="best")
```

```
plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.title('Model Accuracy')

plt.subplot(1, 2, 2)

plt.plot(epochs_range, loss, label='Train Set')

plt.plot(epochs_range, val_loss, label='Val Set')

plt.legend(loc="best")

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.title('Model Loss')

plt.tight_layout()

plt.show()

## VGG16 Model

base_model = VGG16(

    weights='imagenet',

    include_top=False,

    input_shape=(350,350,3)
```

```
)

NUM_CLASSES = 4

vgg_model = Sequential()

vgg_model.add(base_model)

vgg_model.add(layers.Flatten())

layers.Dense(512, activation='relu')

vgg_model.add(layers.Dropout(0.25))

vgg_model.add(layers.Dense(NUM_CLASSES, activation='sigmoid'))

vgg_model.layers[0].trainable = False

vgg_model.compile(

    loss='categorical_crossentropy',

    optimizer='adam',

    metrics=['accuracy']

)

vgg_model.summary()

# Adding Model check point Callback
```

```
mc = ModelCheckpoint(
    filepath="./ct_vgg_best_model.hdf5",
    monitor= 'val_accuracy',
    verbose= 1,
    save_best_only= True,
    mode = 'auto'
    );
call_back = [ mc];
# Fitting the Model
vgg = vgg_model.fit(
    train_data,
    steps_per_epoch = train_data.samples//train_data.batch_size,
    epochs = 50,
    validation_data = val_data,
    validation_steps = val_data.samples//val_data.batch_size,
    callbacks = call_back
    )
```

```
# Loading the Best Fit Model

model = load_model("./ct_vgg_best_model.hdf5")

# Checking the Accuracy of the Model

accuracy_vgg = model.evaluate_generator(generator= test_data)[1]

print(f"The accuracy of the model is = {accuracy_vgg*100} %")

vgg.history.keys()

# Plot model performance

acc = vgg.history['accuracy']

val_acc = vgg.history['val_accuracy']

loss = vgg.history['loss']

val_loss = vgg.history['val_loss']

epochs_range = range(1, len(vgg.epoch) + 1)

plt.figure(figsize=(15,5))

plt.subplot(1, 2, 1)

plt.plot(epochs_range, acc, label='Train Set')

plt.plot(epochs_range, val_acc, label='Val Set')

plt.legend(loc="best")
```

```
plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.title('Model Accuracy')

plt.subplot(1, 2, 2)

plt.plot(epochs_range, loss, label='Train Set')

plt.plot(epochs_range, val_loss, label='Val Set')

plt.legend(loc="best")

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.title('Model Loss')

plt.tight_layout()

plt.show()

## Inceptionv3 Model

from tensorflow.keras.applications.inception_v3 import InceptionV3

base_model = InceptionV3(input_shape = (350, 350, 3),

                         include_top = False,

                         weights = 'imagenet')
```

```
for layer in base_model.layers:
    layer.trainable = False
x = layers.Flatten()(base_model.output)
x = layers.Dense(1024, activation='relu')(x)
x = layers.Dropout(0.2)(x)
# Add a final sigmoid layer with 4 node for classification output
x = layers.Dense(4, activation='sigmoid')(x)
model_incep = tf.keras.models.Model(base_model.input, x)
model_incep.compile(optimizer = tensorflow.keras.optimizers.RMSprop(learning_rate=0.0001),
                    loss = 'categorical_crossentropy',
                    metrics = ['accuracy'])
# Adding Model check point Callback
mc = ModelCheckpoint(
    filepath="./ct_incep_best_model.hdf5",
    monitor= 'val_accuracy',
    verbose= 1,
    save_best_only= True,
```

```
    mode = 'auto'
    );
call_back = [mc];
# Fitting the Model
incep = model_incep.fit(
    train_data,
    steps_per_epoch = train_data.samples//train_data.batch_size,
    epochs = 50,
    validation_data = val_data,
    validation_steps = val_data.samples//val_data.batch_size,
    callbacks = call_back
    )
# Loading the Best Fit Model
model = load_model("./ct_incep_best_model.hdf5")
# Checking the Accuracy of the Model
accuracy_incep = model.evaluate_generator(generator= test_data)[1]
print(f"The accuracy of the model is = {accuracy_incep*100} %")
```

```
incep.history.keys()

# Plot model performance

acc = incep.history['accuracy']

val_acc = incep.history['val_accuracy']

loss = incep.history['loss']

val_loss = incep.history['val_loss']

epochs_range = range(1, len(incep.epoch) + 1)

plt.figure(figsize=(15,5))

plt.subplot(1, 2, 1)

plt.plot(epochs_range, acc, label='Train Set')

plt.plot(epochs_range, val_acc, label='Val Set')

plt.legend(loc="best")

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.title('Model Accuracy')

plt.subplot(1, 2, 2)

plt.plot(epochs_range, loss, label='Train Set')
```

```
plt.plot(epochs_range, val_loss, label='Val Set')

plt.legend(loc="best")

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.title('Model Loss')

plt.tight_layout()

plt.show()

## RestNet50 Model

from tensorflow.keras.applications import ResNet50

base_model = ResNet50(input_shape=(350, 350,3),

                      include_top=False, weights="imagenet",

                      pooling='max')

for layer in base_model.layers:

    layer.trainable = False

model_resnet = Sequential()

model_resnet.add(base_model)

model_resnet.add(Dense(4, activation='sigmoid'))
```

```
model_resnet.compile(optimizer = tf.keras.optimizers.SGD(learning_rate=0.0001),
                     loss = 'categorical_crossentropy',
                     metrics = ['accuracy'])
# Adding Model check point Callback
mc = ModelCheckpoint(
    filepath="./ct_resnet_best_model.hdf5",
    monitor= 'val_accuracy',
    verbose= 1,
    save_best_only= True,
    mode = 'auto'
    );
call_back = [mc];
# Fitting the Model
resnet = model_incep.fit(
    train_data,
    steps_per_epoch = train_data.samples//train_data.batch_size,
    epochs = 50,
```

```
    validation_data = val_data,
    validation_steps = val_data.samples//val_data.batch_size,
    callbacks = call_back
    )
# Loading the Best Fit Model
model = load_model("./ct_resnet_best_model.hdf5")
# Checking the Accuracy of the Model
accuracy_resnet = model.evaluate_generator(generator= test_data)[1]
print(f"The accuracy of the model is = {accuracy_resnet*100} %")
resnet.history.keys()
# Plot model performance
acc = resnet.history['accuracy']
val_acc = resnet.history['val_accuracy']
loss = resnet.history['loss']
val_loss = resnet.history['val_loss']
epochs_range = range(1, len(resnet.epoch) + 1)
```

```
plt.figure(figsize=(15,5))

plt.subplot(1, 2, 1)

plt.plot(epochs_range, acc, label='Train Set')

plt.plot(epochs_range, val_acc, label='Val Set')

plt.legend(loc="best")

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.title('Model Accuracy')

plt.subplot(1, 2, 2)

plt.plot(epochs_range, loss, label='Train Set')

plt.plot(epochs_range, val_loss, label='Val Set')

plt.legend(loc="best")

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.title('Model Loss')

plt.tight_layout()

plt.show()
```

```
## Comparison

algos = ['CNN', 'VGG16', 'InceptionV3', 'Resnet50']

accuracy = [accuracy_cnn, accuracy_vgg, accuracy_incep, accuracy_resnet]

accuracy = np.floor([i * 100 for i in accuracy])

fig = plt.figure(figsize = (10, 5))

# creating the bar plot

plt.bar(algos, accuracy, color ='red', width = 0.3)

plt.xlabel("Algorithms Applied")

plt.ylabel("Accuracy")

plt.show()

### Predictions

def chestScanPrediction(path):

    classes_dir = ["Adenocarcinoma","Large cell carcinoma","Normal","Squamous cell
carcinoma"]

    # Loading Image

    img = image.load_img(path, target_size=(350,350))

    # Normalizing Image

    norm_img = image.img_to_array(img)/255
```

```
    # Converting Image to Numpy Array
    input_arr_img = np.array([norm_img])
    # Getting Predictions
    pred = np.argmax(model.predict(input_arr_img))
    # Printing Model Prediction
    print(classes_dir[pred])
path = "../input/chest-ctscan-images/Data/test/large.cell.carcinoma/000110.png"
chestScanPrediction(path)
tf.keras.models.save_model(model, 'my_model.hdf5')
```

4.2 STREAMLIT IMPLEMENTATION

```
!pip install streamlit
!pip install pyngrok===4.1.1
from pyngrok import ngrok
%%writefile app.py
import tensorflow as tf
import streamlit as st
st.set_option('deprecation.showfileUploaderEncoding',False)
```

```
@st.cache(allow_output_mutation=True)
def load_model():
    model = tf.keras.model.load_model('./ct_resnet_best_model.hdf5')
    return model
model = load_model()
st.write("""
# Lung Cancer Detection """)
file = st.file_uploader("Please upload an Image", type=["jpg", "png"])
import cv2
from PIL import Image,ImageOps
import numpy as np
def import_and_predict(image_data, model):
    size = (350,350)
    image= ImageOps.fit(image_data, size,Image.ANTIALIAS)
    img = np.asarray(image)
    img_reshape = img[np.newaxis,...]
    prediction = model.predict(img_reshape)
```

```
    return prediction

if file is None:

    st.text("Please Upload an Image")

else:

    image = Image.open(file)

    st.image(image,use_column_width=True)

    predictions = import_and_predict(image,model)

    class_names = ["Adenocarcinoma","large  cell  carcinoma","squamous cell carcinoma"]

    string = "The Lung Cancer is type of:" + class_names[np.argmax(predictions)]

    st.success(string)
```

```
path = "../input/chest-ctscan-images/Data/test/large.cell.carcinoma/000110.png"
chestScanPrediction(path,model_incep)
```

```
Large cell carcinoma
```

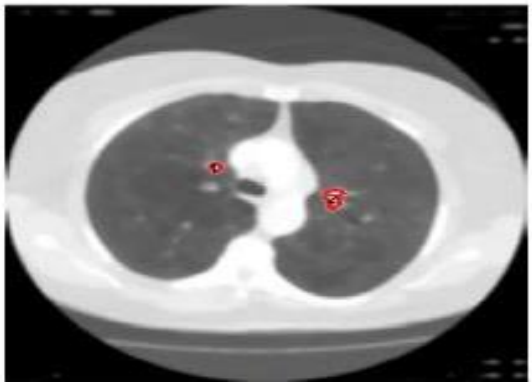

Figure 14: Detection of Lung cancer cells

CHAPTER-05

Conclusion and future work

5.1 CONCLUSION

In the end we conclude that the best current model does not achieve adequate accuracy and does not distinguish the types of cancer obtained. As a result, a new system is offered. The suggested approach uses a ResNet Architecture to detect a lung cancer in a CT scan images.

5.2 Future Work / Enhancements

There is an improvement in accuracy, but it is still not at the best level, which is close to 100 percent and it can render more images in shorter time period and it can supported be on low end GPU also.

5.3 RESEARCH PAPER ACCEPTANCES

Gmail Shivank Srivastava <shivanksrivastava2001@gmail.com>

TURCOMAT ACCEPTANCE

EDITOR TURCOMAT <info.turcomat@gmail.com> Fri, Dec 10, 2021 at 11:40 AM
To: er.kartik18102016@gmail.com, shivanksrivastava2001@gmail.com, aanchal.vij@galgotiasuniversity.edu.in

We suggest Login to Turcomat website https://turcomat.org/ with your Login Credentials and Conform your acceptance

Dear Author

----------Reviewer Evaluation ----------

1. Originality: 70%
2. Scope of Article: 90%
3. Creative: Yes
4. Understandable: Yes
5. References: Cited Properly
6. Results: Satisfactory

Final Decision: Accepted

Acceptance Confirmation

I am pleased to inform you that your manuscript Titled " Lung cancer detection using scan images " is accepted for publication in the Turkish Journal of Computer and Mathematics Education. ISSN 1309-4653 | **Founded:** 2009 .

For any further query feel free to contact us.
APC/Open access charges of the article is 100 USD./ 7,000 INR (only for India payments) Please pay at the below mentioned link.

Link: https://ecaas.traknpay.in/view.php?id=13408 (Indian author)

RazorPay Link : https://rzp.io/i/rDWVycS (foreign author)

Note: Resubmit your article here in this mail in Word Format along with payment receipt.

Upon sending the receipt of payment, the article would be online within 15 days.
I would like to remind you that you could send your future manuscripts to the Turkish Journal of Computer and Mathematics Education.

Sincerely yours,
Editor
Turkish Journal of Computer and Mathematics Education
info.turcomat@gmail.com

https://turcomat.org/index.php

--

Thanks

TURCOMAT Team

Link: https://turcomat.org/index.php

5.4 REFERENCES

[1]. Gindi,A. M., Al Attiatalla, T. A., & Sami, M.M. (2014)

"A Comparative Study for Comparing Two Feature

Extraction Methods and Two Classifiers in Classification of

Earlystage Lung Cancer Diagnosis of chest x-ray images." Journal of American Science, 10(6): 13-22.

[2]. Suzuki, K., Kusumoto, M., Watanabe, S. I., Tsuchiya, R., & Asamura, H. (2006) "Radiologic classification of small adenocarcinoma of the lung: radiologic-pathologic correlation and its prognostic impact," The Annals of Thoracic Surgery. 81(2): 413-419.

[3]. Xiuhua,G., Tao, S., & Zhigang, L.(2011) "Prediction Models for Malignant Pulmonary Nodules Based-on Texture Features of CT Image." In Theory and Applications of CT Imaging and Analysis. DOI: 10.5772/14766.

[4] Aggarwal, T., Furqan, A., & Kalra, K. (2015) "Feature extraction and LDA based classification of lung nodules in chest CT scan images." 2015 International Conference On Advances In Computing, Communications And Informatics (ICACCI), DOI: 10.1109/ICACCI.2015.7275773.

[5]. Jin, X., Zhang, Y., & Jin, Q. (2016) "Pulmonary Nodule Detection Based on CT Images Using Convolution Neural Network." 2016 9Th International

Symposium On Computational Intelligence And Design (ISCID). DOI: 10.1109/ISCID.2016.1053.

[6]. Sangamithraa, P., & Govindaraju, S. (2016) "Lung tumour detection and classification using EK-Mean clustering." 2016 International Conference On Wireless Communications, Signal Processing And Networking (Wispnet). DOI: 10.1109/WiSPNET.2016.7566533.

[7]. Roy, T., Sirohi, N., & Patle, A. (2015) "Classification of lung image and nodule detection using fuzzy inference system." International Conference On Computing, Communication & Automation. DOI: 10.1109/CCAA.2015.7148560.

[8]. Ignatious, S., & Joseph, R. (2015) "Computer aided lung cancer detection system." 2015 Global Conference On Communication Technologies (GCCT), DOI: 10.1109/GCCT.2015.7342723.

[9]. Rendon-Gonzalez, E., & Ponomaryov, V. (2016) "Automatic Lung nodule segmentation and classification in CT images based on SVM." 2016 9Th International Kharkiv Symposium On Physics And Engineering Of Microwaves, Millimeter And Submillimeter Waves (MSMW). DOI: 10.1109/MSMW.2016.7537995.

[10]. Miah, M.B.A., & Yousuf, M.A. (2015) "Detection of lung cancer from CT image using image processing and neural network." 2015 International Conference on Electrical Engineering and Information Communication Technology (ICEEICT): 1-6.