MEDICAL IMAGE CLASSIFICATION USING DEEP CONVOLUTIONAL NEURAL NETWORKS FOR DISEASE DIAGNOSIS

K.Sreedevi *20951A12A1*

Nori Pavana Akhila *20951A1255*

Chokka Prabhu Sandeep *20951A1258*

MEDICAL IMAGE CLASSIFICATION USING DEEP CONVOLUTIONAL NEURAL NETWORKS FOR DISEASE DIAGNOSIS

A Project Report

submitted in partial fulfillment of the

requirements for the award of the degree of

Bachelor of Technology

in

Information Technology

By

K.Sreedevi	**20951A12A1**
Nori Pavan Akhila	**20951A1255**
Chokka Prabhu Sandeep	**20951A1258**

DEPARTMENT OF INFORMATION TECHNOLOGY

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad – 500 043, Telangana

April, 2024

DECLARATION

We certify that

a. The work contained in this report is original and has been done by us under the guidance of our supervisor(s).
b. The work has not been submitted to any other Institute for any degree or diploma.
c. We have followed the guidelines provided by the Institute in preparing the report.
d. We have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
e. Whenever we have used materials (data, theoretical analysis, figures, and text) from othersources, we have given due credit to them by citing them in the text of the report and giving their details in the references. Further, we have taken permission from the copyright owners of the sources, whenever necessary.

Place: Dundigal

Date: 20/4/2024

Signature of the Student

Roll No. 20951A12A1 Sreeja

20951A1255 Archana

20951A1258 Sandeep

CERTIFICATE

This is to certify that the project report entitled **Medical image classification using deep convolutional neural networks for disease diagnosis** submitted by **Ms. K. Sreedevi, Ms. Nori Pavana Akhila, Mr. Chokka Prabhu Sandeep** to the Institute of Aeronautical Engineering, Hyderabad in partial fulfillment of the requirements for the award of the Degree Bachelor of Technology in **Information Technology** is a bonafide record of work carried out by their under my guidance and supervision. The contentsof this report, in full or in parts, have not been submitted to any other Institute for the award of any Degree.

Supervisor

Dr. M. Purushotham Reddy

Head of the Department

Dr. M. Purushotham Reddy

Head of the Department
Information Technology
INSTITUTE OF AERONAUTICAL ENGINEERING
Dundigal, Hyderabad - 500 043

Date: 20/4/2024

APPROVAL SHEET

This project report entitled **Medical image classification using deep convolutional neural networks for disease diagnosis** by **K. Sreedevi, Nori Pavana Akhila, Chokka Prabhu Sandeep** is approved for the award of the Degree Bachelor of Technology in **Information Technology.**

Examiner

Supervisor
Dr. M. Purushotham Reddy

Principal
Dr. L V Narasimha Prasad

PRINCIPAL
INSTITUTE OF AERONAUTICAL ENGINEERING
Dundigal, Hyderabad - 500 043
Telangana State

Date: 20/11/2024

Place: Dundigal

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts with success.

We thank our college management and **Sri. M Rajashekar Reddy**, Chairman, for providing us the necessary infrastructure to carry out the project work.

We express our sincere thanks to **Dr. L.V. Narasimha Prasad**, beloved Principal who has been a great source of information for our work and **to Dr. M. Purushotham Reddy**, Professor and Head, Department of Information Technology, for extending his support to carry out this project work.

We are especially thankful to our internal supervisor **to Dr. M. Purushotham Reddy**, Professor and Head, Department of IT, for their internal support and professionalism who helped me in shaping the project into successful one. A special thanks to our beloved Project Coordinator **Dr. U Shivaji,** Associate Professor, Department of Information Technology.

We take this opportunity to express our thanks to one and all who directly or indirectly helpedme in bringing this effort to present form.

K. Sreedevi 20951A12A1

Nori Pavana Akhila 20951A1255

Chokka Prabhu Sandeep 20951A1258

ABSTRACT

The medical image classification is a significant factor in modern medical arena as the wide range of diseases and conditions can be detected and confirmed at early stages. Convolutional Neural Networks (CNNs) has become a prominent contribution in the sphere of automating analysis of medical images, and performing this for the most part, the accuracy and efficiency is very high. This subsection delineates the major points involving medical image classification using CNNs in the research. The main goal of this research is to construct and assess CNN-based algorithms for the section of medical pictures into known groups. From different modality images including X-rays, MRI scans, CT scans and histopathological slides these cover a plenty area. In this process the modules are able to categorize domains into normal vs abnormal, benign vs. cancerous, for example, or in a particular disease type. Reading and analyzing articles in healthcare is an integral component of the field. The methodology of the research, consisting in a number of steps, is applied here in. Firstly, the dataset of annotated medical images is gathered and data preprocessing is carried out so as to avoid data inconsistency. The outermost layer of a CNN model is set, including a convolution layer and two pooling & pooling layers. Transfer learning methods can be used, for example, through pre-trained models such as VGG, ResNet, and Inception which have bigger chances of rising model performance. Training and testing are done with a split-off sample of data, while the performance of the trained model is measured on an independent or separate test set of data. Evaluation metrics for this explication include precision, recall, accuracy, F1-scores, and the area under the receiver operating characteristic curve(AUC-ROC) metrics. Besides, approaches such as data augmentation and regularization, which conform the re-learning, are also used to avoid the phenomenon of overfitting.

Keywords: Reverse transcription Polymerase chain reaction (RT-PCR), Diagnostic tools, Chest X-ray,Deep learning models, CNN, VGG19, U-Net, Lung segmentation.

CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Introduction:

The sentence indicated that a project begins with a background section which lays the emphasis on medical image analysis that has grown in healthcare in recent years. It amounts to how accurate disease diagnosis represents an important part of healthcare device, the problems encountered by specialists who read medical images in doing so, and how those problems can be potentially solved with CNN which performs better than human in similar task. It enlists the project aims and gives a brief to describe the methods used.

Deep learning based CNNs in medical image classification can be thought of as a revolutionary discovery offering diagnostic assistance findings and improvement in healthcare. The advancement of the CNN in the medical science fields make it a very useful tool to help in analyzing and interpreting the images of this data, therefore all the medical images such as MRIs, CTs, ultrasounds etc. will be able to detect a disease accurately and timely.

It utilizes the strength of one of the deep learning technique class types called Convolution Neural Network (CNN) which has the ability of automatically learning and extracting the complicated patterns and the characteristics from pictures. Through the application of numerous convolutional layers, these networks can tell differences between very specific features within medical images, including but not limited to X-rays, MRIs, and CT scans, and being able to perform significant diagnostics more accurately than humans can. By the adoption of CNN in the medical image analysis, there has been a substantial improvement in the diagnostic accuracy and quick execution of images. It facilitates the process of distinguishing and separating abnormalities, the occurrence of the tumors, lesions, and other pathological conditions. Moreover, assisting the healthcare professionals with the qualitative evaluation, reducing the uninterpretation time, and probably the diagnostic errors are what are achieved by these systems.

1.2 Existing System

Deep learning based on convolutional neural networks (CNNs) has now emerged as the most applicable tools for recognizing disease in a certain condition as they are good at discerning complex features found in such images. Broadly speaking, the general schemes imply the implementation of pretrained architectures such as ResNet and VGG, and thereafter fine-tuning the networks utilizing medical imaging datasets. Expanding to other domains, transfer learning methods becomes critically more important as useful features are now extracted from the data. On the other hand, data augmentation techniques are used to provide solutions to the challenge of limited data sets, hence ensuring the models are efficient and offers an equal level of generalization.

Complementing this utilization, attention mechanisms and ensemble learning are also applied to further boost the performance of these models. The combination of attention mechanisms allows the network to assign importance to the prominent information in the images hence enhancing the discriminative capabilities of the model. This is outlined in an example of how a diverse group of models are fused using ensemble learning techniques to provide higher accuracy and dependability in classification outcomes.

By these multidimensional strategies, our deep CNNs facilitate the exact determination of diseases in the medical images, which are further conducted to early diagnosis and the conduction of the optimal treatment. Healthcare practices will be able to shorten the time needed for diagnosis and increased accuracy when considering the treatment course, using deep learning and transfer learning much more efficiently. It is not only a benefit for patients in this way, but also helps to improve the work patterns of doctors and nurses as it frees up some time so that they can concentrate on more important matters

1.2.1 Limitations of the Existing System:

- **Limited Generalization**: ResNet or VGG-16 which are saturated deep learning architectures do not often transfer the extracted features on datasets that have specific and rare diseases.
- **Data Heterogeneity**: Datasets used in medical imaging science often have high variance in imaging quality, resolution, and conduction protocols. The pre-trained models that are fine-tuned on one dataset and then introduced another may fail the

operational effectiveness due to the variant of the heterogeneity.

- **Domain Shift**: The key idea of transfer learning is that data distributions between the source domain (for training in the medical imaging area) and the target domain (medical imaging) are similar. On the bright side, the models demonstrated good transferability between datasets, but severe domain shifts in medical imaging datasets can affect model's performance.
- **Limited Data Availability**: Although the data augmentation methodology is able to overcome size problems with medical imaging datasets, rare diseases or some imaging modalities may still remain problematic because of the limitation of data particular to the modality in question and the fact that data covering this domain is not large enough.
- **Interpretability**: Deep CNNs are usually recognized as black boxes, hence non-expert doctors cannot understand and trust their diagnosing decisions, which (thisinformation) is an obstacle to the system`s adoption. Thus, the resistance to adopt the automated diagnosis systems comes to be evident
- **Overfitting**: Data augmentation methods might not completely protect neural networks from overfitting the case when working with dataset of small or skewed structure. These could just bring undesirable results to the model in question.
- **Computational Resources**: It is very hard to train such deep CNNs, and that takes a huge lot of computational resources and time, which is probably a problem that can make deploying them in settings with a little resource a bit difficult.
- **Risk of Misdiagnosis**: Regarding the automated diagnosis systems for diseases, which are based on deep learning, though the technology has improved still, those systems have a potential of misdiagnosing cases; hence specialists may overlook their cases and wrong treatment plan made to patients.

1.3 Proposed System

The proposed solution delineates the methodology and approach embraced in this endeavour. It intricately describes the employment of deep CNN architectures to extract features and classify diseases from medical images. The discussion encompasses the selection and preparation of datasets for training and validation, the application of preprocessing methods, the design of model architecture, training strategies, and the metrics employed to evaluate the model's efficacy. Furthermore, it underscores the prospective

benefits and positive outcomes associated with the utilization of deep learning techniques in medical image classification for disease diagnosis.

1.3.1 Advantages of the Proposed System:

- Convolutional Neural Networks (CNNs) demonstrate proficiency in discerning intricate patterns, facilitating precise identification of diseases from medical images.
- The utilization of pre-trained models expedites training processes and enhances performance, particularly when confronted with limited medical datasets.
- CNN architectures autonomously extract pertinent features, thereby assisting in disease characterization and classification.
- Precise classification fosters early detection of diseases, a critical factor in initiating timely treatment and improving prognoses.
- CNN-based models hold promise in automating diagnostic procedures, thus reducing healthcare workloads and bolstering overall efficiency.
- Post-training, these models have the capacity to scale across various diseases, optimizing diagnostic workflows across diverse medical scenario.

CHAPTER 2

LITERATURE SURVEY

2.1 Literature Review

A paper titled "Automatic segmentation of cerebral MR images using artificial neural networks," by J. Alirezaie, M. E. Jeraining, and C. Nahmias [1], was published in the journal IEEE Transactions on Nuclear Science in 1998, where authors discuss applying ANNs to the automatic segmentation of cerebral MR images. The fundamental objective is to demonstrate the approach of ANNs, that are the computational models inspired by the human brain's neural network to classify the cerebral MR images as different regions or structures. The article probably focuses only on the exact architecture along with structure of the neural network, which will be employed in segmentation. The lighting layer counts, loop configuration, and training strategy will be described as well. The achievement obtained from applying ANN to the segmentation duty would be explained with the help of qualitative metrics to measure how efficient each step is. The epilogue will address the obtained results, discuss their value and even point out the directions for further research on the basis of the obtained results. The paper generally presents the robotic way of sectioning out the MR images within the medicine area by involving the ANNs, thus exploration the intelligence of the neural networks in this field.

S. Belleville, C. Fouquet, S. Duchesne, D.L. Collins, and C. Hudon [2], titled "Detecting early preclinical Alzheimer's disease via cognition neuropsychiatry and neuroimaging: The paper titled "Current quantitative techniques and future recommendations for testing," published in the Journal of Alzheimer's Disease in 2014, contains a detailed qualitative review of methods for the early preclinical stage detection of AD. The paper generally spotlights applying cognitive tests, psychiatrists' observations, and neuroimaging for AD presymptomatic diagnosis. The authors may possibly explain several theories and what had been observed during the research that helped for recognizing the early cognitive changes, psychiatric symptoms, and neuroimaging finding which may be early markers of AD. Types of tests that can be conducted and which strategies could combine information from different modalities such as cognation, neuropsychiatry and neuroimaging may be recommended. The term "qualitative review"

helpful for clinicians, researchers and other professionals involved in the Alzheimer's research or diagnosis.

S. Wang, Z. You, L. Guo, P. Fillip and T. Flore. Yuan [3], a paper called “Three-dimensional displacement field estimation in structural MRI: Towards early Alzheimer’s disease detection,” triggered a breakthrough in 2016, and has been published in the Journal of Alzheimer’s Disease. This paper presented a new AD detection approach. This technique involves LP building 3D displacement field estimations from the structural MRI data. Through spatial displacement pattern analysis, the authors want to see the structural changes accompanying the AD. This method of analysis is a possible innovation tool of non-invasive AD detection. With this innovative technique, we are expecting progress in early diagnosis and an understanding of the structural alterations in the brain related to AD.

T. Altaf, S. Anwar, N. Gul, N. Majeed, and M. Majid work [4] together in the FTC 2017; they enjoyed presenting the multiclass approach of Alzheimer's disease classification. The mentioned authors make use of the hybrid abilities, probably exploiting the combination of different data sources and feature extraction techniques, for an accurate AD classification. The question of the paper's multiclass frame is how to disentangle the various stages or classes of the AD. Thus, the FTC meeting in November 2017, which addressed this research project, was not only aimed at classifying the AD, but also at making it possible to detect different stages of the disease early and in a timely manner.

Developed by Z. Lao, D. Shen, Z. Xue, B. Karacali, S.M. Resnick and C. Davatzikos [5], in the article in neuroimage "Morphological classification of brains via high-dimensional shape transformations and machine learning methods," the authors provide the novel method for morphological classification of brains. The study accomplishes this goal via formation of the 3-Dimensional shapes by corresponding transformations and Machine Learning Techniques, which categorize brain morphologies. The main purpose authors use simulation in this field is to have a better view of the brain structure changes. Coordinating the shape modifications and machine learning would facilitate accurate and automated brain morphology classification; which can be useful for research on the central nervous system and application in medical practice.

G. Fung and J. Stoeckel [6], titled, “An SVM based classifier for the AD images classification using image spatial information,” published in the Knowledge and Information Systems, 2007, the authors develop the SVM feature selection method for the classification of structural SPECT images of AD. In the SPECT images, Spatial Information is being brought into the feature selection process via identifying hotspots extracted from the images. The study adaptation of SVM for spatial details of SPECT images contributes to the improvement of accurateness of AD classification and illustrates the importance of particulars in neuro imaging. The paper is exactly some of the clinical approaches that use neuroimaging to determent Alzheimer's Disease.

Chincarini, P. Bosco, P. Calvini, G. Gemme, M. Esposito, C. Olivieri, et al. [7], titled "Local MRI analysis approach in the diagnosis of early and prodromal Alzheimer’s disease," published in NeuroImage in 2011, introduces a localized MRI analysis method for diagnosing early and prodromal stages of Alzheimer's disease (AD). The study focuses on detailed regional analysis using MRI data, aiming to identify subtle structural changes associated with AD at early stages. By employing a local analysis approach, the research contributes to improving the sensitivity of MRI-based diagnostics for early and prodromal Alzheimer's disease detection.

E. Westman, L. Cavallin, J.-S. Muehlboeck, Y. Zhang, P. Mecocci, B. Vellas, et al. [8], titled "Sensitivity and specificity of medial temporal lobe visual ratings and multivariate regional MRI classification in Alzheimer’s disease," published in PLoS ONE in 2011, the authors investigate the diagnostic accuracy of two distinct approaches for detecting Alzheimer's disease (AD). The study evaluates the sensitivity and specificity of visual ratings of the medial temporal lobe and a multivariate regional MRI classification method. The research aims to assess the reliability of these techniques in distinguishing AD from non-AD cases. The findings contribute to understanding the effectiveness of visual assessments and multivariate regional MRI analyses in Alzheimer's disease diagnosis, providing insights into the strengths and limitations of each approach.

O.B. Ahmed, J. Benois-Pineau, M. Allard, C.B. Amar, and G. Catheline, who classify Alzheimer’s disease (AD) subjects from visual features extracted from hippocampal regions in magnetic resonance imaging (MRI) by [9], are the most appreciated. Their approach is

found in Multimedia Tools and Applications. Identifying the distinctive features of the hippocampus, the study considers the classification methods to discriminate between the groups with and without AD. Through such approach, the scientific field of AD diagnosis is able to evaluate the specific hippocampal features in order to refine the classification accuracy using neuroimaging data.

2.1 Requirement Specifications

2.1.1 Hardware Requirements:

- Processor - Pentium –IV
- RAM - 4 GB (min)
- Hard Disk - 20 GB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- Monitor - SVGA

2.1.2 Software Requirements:

- Operating system : Windows 7 Ultimate.
- Coding Language : Python.
- back-End : Python.

2.2 System Study

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

2.2.1 Economical Feasibility

This report studies the economic values of such disruption within the organization. The significant limitations on research and development funding imposed by the budget

cycle entail making the necessary justifications for the spending. Subsequently, the budget - providing system was configured to function within the allotted funds. Perhaps the most significant factor that helps keep our services affordable is the fact that majority of the technologies we use are all open-source and this further cut our costs. Broader distribution and spawning forced diversion of the fleet from essential cargo procurement. The project was able to achieve this by using really available technologies and carefully planning which solutions should be purchased. Consequently, of course the budget was not exceeded. With this system, the fiscal costs are minimized, performance is enhanced and development of such applications are made cost effective. Additionally, the study focuses on correctly allocating funds to critical components that facilitate leveraging of cost-effective open- source resources in a bid to guarantee maximum outcomes for the system. Proper financial stewardship and emphasis on value-creating investments foundations are established throughout the project with this aim set to generate the maximum amount of profit without the ever-increasing expenses. Overall, this model of financial logistics conforms to the standard of financial responsibility for the organization and apt observation of resources management for sustainable and financially viable development.

2.2.2 Technical Feasibility

A research project is carried out to determine the possibility of introduction of this system, with complete attention to all its technical features. It is very important for any system that we develop would not actually be too much for the client with the limited available resources of technical type, and thus it is desirable to keep that system easy for the user. This mean it is important that the developed system has lightweight requirements and can activate them without much reconstruction during the implementation phase. This measure offers insurance that the technical requirements of the system remain within manageable limits, therefore making system disruption equitable to system failure. This strategy aims to ensure this phase is as effective and resource- saving as possible, and is integrated smoothly with the rest, thus, reducing the costs and adding to complexity. The modesty of technical requirements is a key element in carrying out this objective because the solution should be discreet enough to be interoperable without posing overwhelming burdens to the existing technical system. Ultimately, there is less fragmentation and more than likely better acceptance, both of which are important factors in the development of a successful and effective system.

2.2.3 Social Feasibility

In this part, I will try to accomplish the goal of apprising the user acceptance of the system, which covers the user training process to efficiently use the system fully and

comprehensively. In this sense, the liability of the system on users shouldn't be perceived as a threat but regarded as a helpful instrument. Acceptance of users depends on how adjustable the methods that provide necessary information to users with the aid of which they become familiarized with the system are. Building users' level of trust is poor but not just that it's the ones who use the product to the end that are able to give positive feedback, hence, its which they are the ones who give the accurate details and hence it's built upon making them feel safe. Such an approach places high value on framing the user experience in positive terms and aims at the engagement of the end-users and creation of a collaborative platform. Through helping users with tools and resources on how the system works users will be able obtain skills, and knowledge that go towards overcoming the reluctance in acceptance. Additionally, the promotion of the users' responding positively to the questions will bring about am interactive environment where mistakes and complaints are dealt with immediately and a culture of the system’s capacity to meet the needs of the customers is developed. User acceptance in the end is where the system’s success lies as it brings the reality of sustainability through its thoughtful and interactive training methodologies

CHAPTER 3
DESIGN

3.1 UML Diagrams

UML stands for Unified Modeling Language or SFL (Standards for Facilities). UML is a general-purpose object-orientated modeling language that has become popular in the field of software engineering and is based on standard principles. Control is done, and it is also created by, that particular group. The Object Management Group.

- The intention is that UML models become a universal standard for the creation of object-centric software diagrams. In its current form UML is comprised of two major components: finally, we will create a Meta model and a notation. of this, we will Model our content Meta and the notation. In the future, UML can also be supported by a method or mechanism that is either related to or connected with UML by some means.
- Unified Modeling Language is standard language for modeling unambiguously replacing the descriptions which are visual, explicit and also documentation ones of software through the use of the artifacts. The application of the modeling includes business modeling as well as non-software systems.
- The UML reflects a chosen set of best engineering practices that have been effectively employed to model of an extensive software system.

3.1.1 Use Case Diagram:

In the Unified Modelling Language (UML), a use case diagram is a behavioural diagram, which stands for and comes from the analysis involving and of use cases. Its main job is to draw a diagram which symbolizes how a system performs various tasks. It iconizes actors, their use cases and depicts the cause-effect relationships between those use cases. The primary value of the use case diagram is that it displays the operation of system actors for which functions are composed. By depicting actors' roles in the system as shown in Fig. 3.1.1, the model could be seen as in the image below.

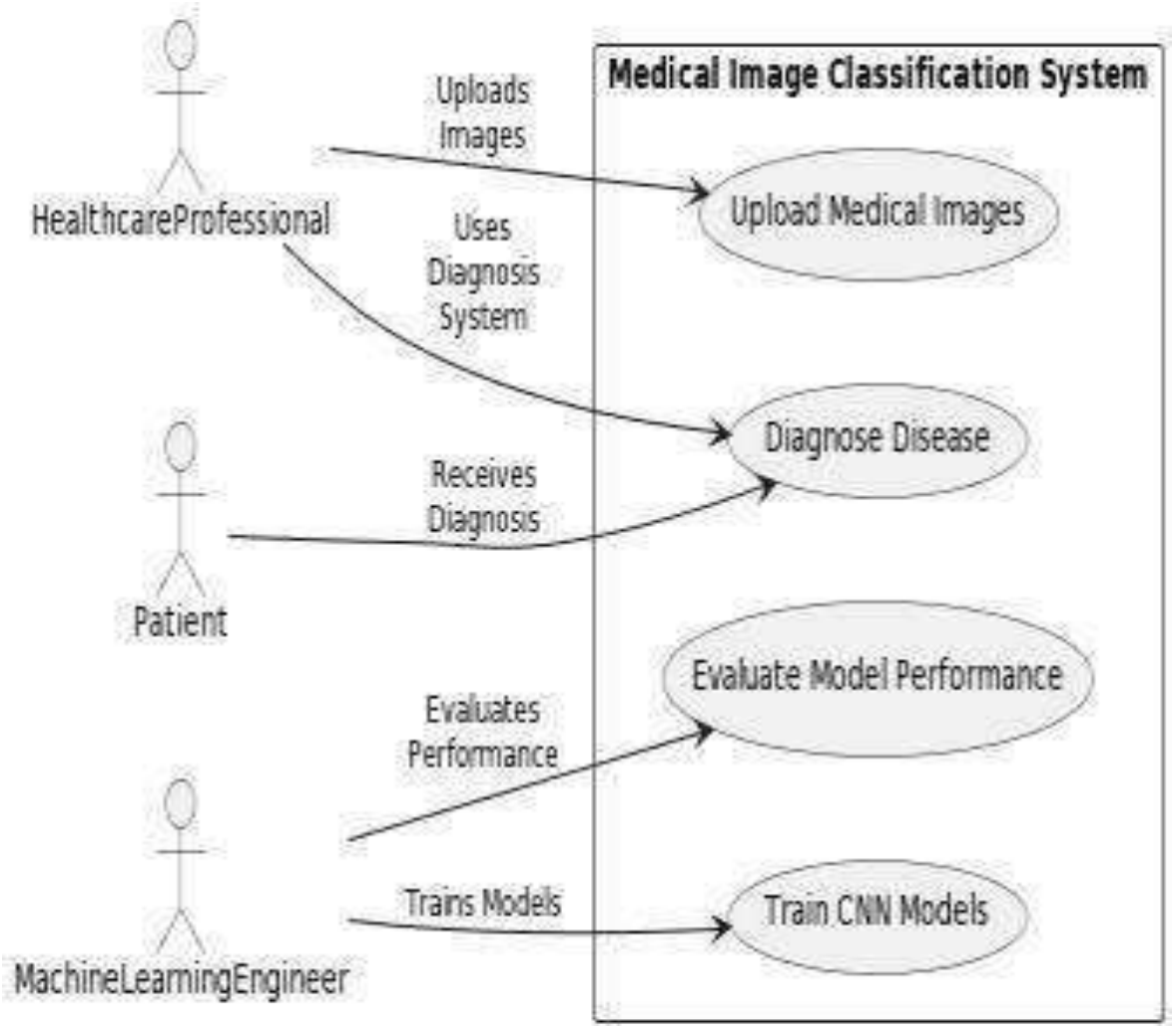

Fig. 3.1.1 Use Case diagram

3.1.2 Class Diagram:

In software engineering, a class diagram in which the unifier model language (UML) is a sort of static structural diagram that is used in describing the structure of a system by showing the ins system classes and their attributes, the operations (or methods) and the relations among them. Through this, maintains the information content class hierarchy of the shown figure in Fig. 3.1.2.

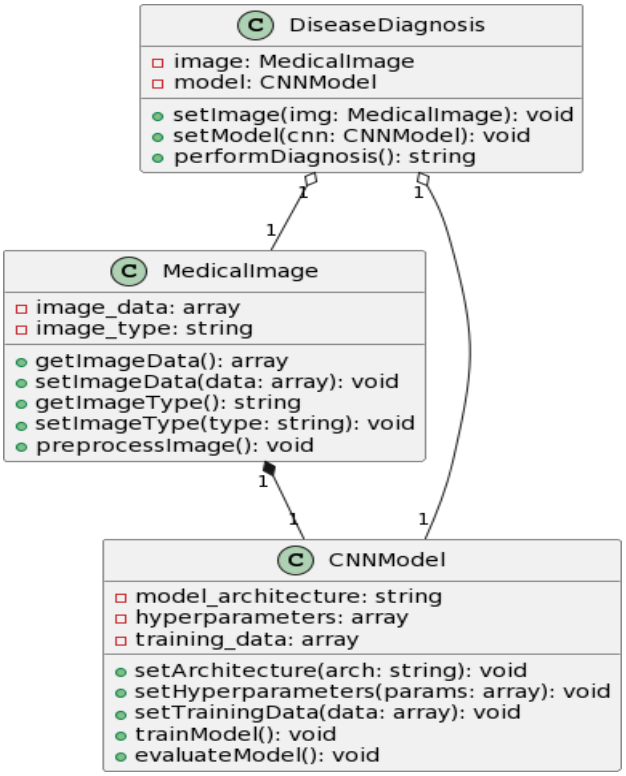

Fig. 3.1.2. Class diagram

3.1.3 Sequence diagram:

A sequence diagram among the diagrams that use the Unified Modeling Language (UML) in the interaction diagram category shows how processes run in order and with whom. It is an XBee stack communication protocol which is represented in Fig 3.1.3. Timing diagrams are also referred to as Event DAG, event sequences, and scenario diagrams.

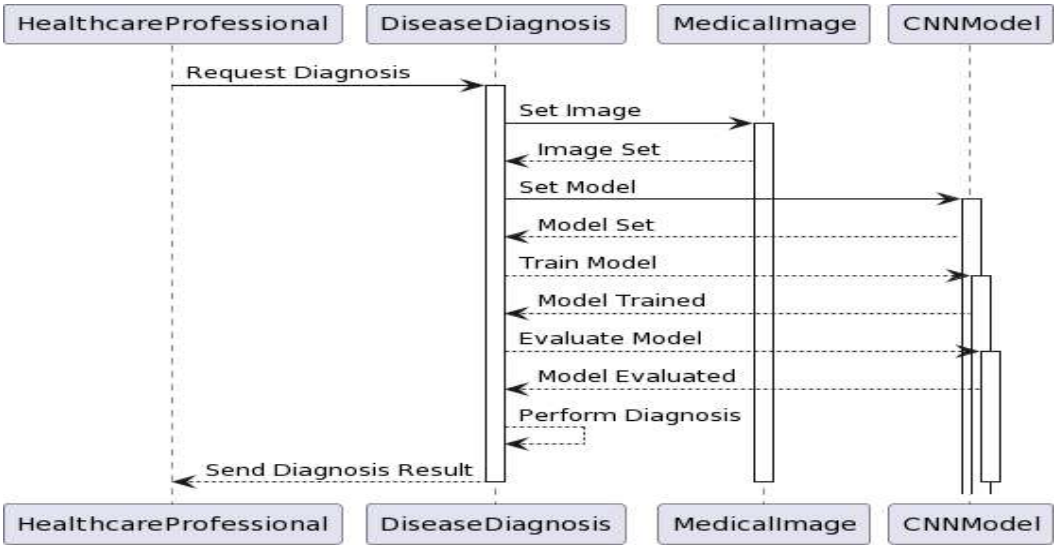

Fig. 3.1.3. Sequence diagram

3.1.4. Activity Diagram:

Activity diagrams in medical image classification, as an example of visual representation of all the sequential and parallel pathways, involve the image processing steps. In the medical image classification, processing diagrams are very important, it helps in the process of showing the step-by-step sequence how data is acquired, processed and finally classified. The diagram will often be composed of vital steps including image preprocessing, feature extraction, model training and validation, as well as testing. It shows how the information flows inside the decision structure along with the information processing pipeline in a graphical representation as demonstrated in the figure below. Fig 3.1.4.

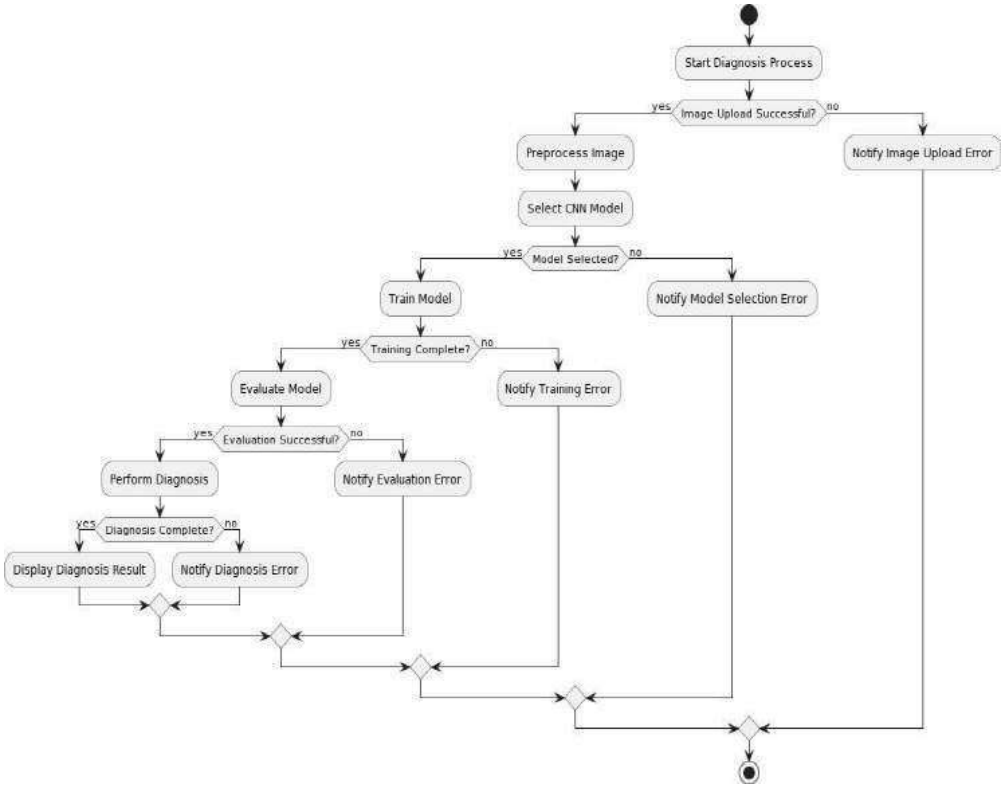

Fig. 3.1.4. Activity diagram

CHAPTER 4

METHODOLOGY AND IMPLEMENTATION

4.1 Modules:

- Install Tensorflow
- Install Numpy
- Install Pandas
- Install Matplotlib
- Install Scikit – learn

4.2 Module Descriptions:

4.2.1 Data Collection and Preprocessing Module:

- **Data Gathering**: Collect and curate a diverse dataset of medical images related to various diseases.
- **Data Preprocessing**: Standardize, clean, and preprocess the images to ensure uniformity, proper formatting, and resolution for model input.

4.2.2 Model Training and Validation Module:

- **Training Setup**: Train the CNN models using the prepared dataset, employing techniques like data augmentation to improve model generalization.
- **Validation and Hyperparameter Tuning**: Validate models on a separate dataset, fine-tune-parameters, and optimize model performance.

4.2.3 Model Evaluation and Interpretation Module:

- **Performance Metrics Calculation**: Evaluate model performance using metrics like accuracy, precision, recall, and F1-score to assess classification effectiveness.
- **Confusion Matrix Analysis**: Analyze the confusion matrix to understand the model's strengths and weaknesses in disease classification.

4.3 Python Implementation

4.3.1 What is Python:

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard libraries which can beused for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

4.3.2 Advantages of Python:

Extensive Libraries: Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

1. Extensible: As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.
2. Improved Productivity: The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.
3. Embeddable: Complimentary to extensibility, Python is embeddable as well. You can

put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4.4 Source Code:

The given below is the python implementation for the project

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
import matplotlib.pyplot as plt
from tkinter import ttk
from tkinter import filedialog
import pandas as pd
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
import pickle
import os
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
import seaborn as sns
```

```
import cv2

from keras.utils.np_utils import to_categorical

from keras.layers import  MaxPooling2D

from keras.layers import Dense, Dropout, Activation, Flatten, GlobalAveragePooling2D, BatchNormalization

from keras.layers import Convolution2D

from keras.models import Sequential, load_model, Model

from keras.applications import VGG16

from keras.applications import VGG19

from keras.applications import DenseNet201

from keras.applications import ResNet50

from sklearn.metrics import precision_score

from sklearn.metrics import recall_score

from sklearn.metrics import f1_score

from sklearn.metrics import accuracy_score

from sklearn.model_selection import train_test_split

from keras.callbacks import ModelCheckpoint

import keras

from sklearn.metrics import accuracy_score

from sklearn.model_selection import train_test_split

from keras.applications import InceptionV3

main = Tk()

main.title("Medical Image Classification using Deep Convolutional Neural Networks for Disease Diagnosis")

main.geometry("1300x1200")

global filename
```

```
global dataset
global X, Y
global X_train, X_test, y_train, y_test, labels
global accuracy, precision, recall, fscore, vgg19_model
labels = []
for root, dirs, directory in os.walk("Dataset"):
    for j in range(len(directory)):
        name = os.path.basename(root)
        if name not in labels:
            labels.append(name.strip())
def getLabel(name):
    index = -1
    for i in range(len(labels)):
        if labels[i] == name:
            index = i
            break
    return index
#fucntion to upload dataset
def uploadDataset():
    global filename, filename, X, Y
    text.delete('1.0', END)
    filename = filedialog.askdirectory(initialdir=".") #upload dataset file
    text.insert(END,filename+" loaded\n\n")
    if os.path.exists('model/X.txt.npy'):
        X = np.load('model/X.txt.npy')
```

```
    Y = np.load('model/Y.txt.npy')
else:
    X = []
    Y = []
    for root, dirs, directory in os.walk(filename):
        count = 0
        for j in range(len(directory)):
            if count < 1500:
                name = os.path.basename(root)
                if 'Thumbs.db' not in directory[j]:
                    img = cv2.imread(root+"/"+directory[j])
                    img = cv2.resize(img, (32,32))
                    X.append(img)
                    label = getLabel(name)
                    Y.append(label)
                    print(name+" "+str(label)+" "+str(count))
                    count = count + 1
    X = np.asarray(X)
    Y = np.asarray(Y)
    np.save('model/X.txt',X)
    np.save('model/Y.txt',Y)
text.insert(END,"Total images found in dataset : "+str(X.shape[0])+"\n")
text.insert(END,"Diseases Found in Dataset : "+str(labels))
unique, count = np.unique(Y, return_counts=True)
height = count
```

```
    bars = labels
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.xlabel('Disease Type')
    plt.ylabel('Count')
    plt.title("Dataset Class Labels Graph")
    plt.show()
def preprocess():
    text.delete('1.0', END)
    global X_train, X_test, y_train, y_test, X, Y
    X = X.astype('float32')
    X = X/255
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices)
    X = X[indices]
    Y = Y[indices]
    Y = to_categorical(Y)
    img = X[0]
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
    text.insert(END,"Dataset Image Processing & Normalization Completed\n\n")
    text.insert(END,"Total Images found in dataset : "+str(X.shape[0])+"\n")
    text.insert(END,"Total features found in each Image : "+str((X.shape[1] * X.shape[2] *
X.shape[3]))+"\n\n")
    text.insert(END,"80% dataset records used to train ML algorithms :
"+str(X_train.shape[0])+"\n")
```

```
    text.insert(END,"20% dataset records used to train ML algorithms : "+str(X_test.shape[0])+"\n")

    text.update_idletasks()

    X_test = X_train[0:200]

    y_test = y_train[0:200]

    cv2.imshow("Processed Image", cv2.resize(img, (300, 300)))

    cv2.waitKey(0)

def calculateMetrics(algorithm, predict, y_test):

    a = accuracy_score(y_test,predict)*100

    p = precision_score(y_test, predict,average='macro') * 100

    r = recall_score(y_test, predict,average='macro') * 100

    f = f1_score(y_test, predict,average='macro') * 100

    accuracy.append(a)

    precision.append(p)

    recall.append(r)

    fscore.append(f)

    text.insert(END,algorithm+" Accuracy :  "+str(a)+"\n")

    text.insert(END,algorithm+" Precision : "+str(p)+"\n")

    text.insert(END,algorithm+" Recall    : "+str(r)+"\n")

    text.insert(END,algorithm+" FScore    : "+str(f)+"\n\n")

    text.update_idletasks()

    conf_matrix = confusion_matrix(y_test, predict)

    plt.figure(figsize =(6, 5))

    ax = sns.heatmap(conf_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="viridis" ,fmt ="g");

    ax.set_ylim([0,len(labels)])
```

```
    plt.title(algorithm+" Confusion matrix")
    plt.ylabel('True class')
    plt.xlabel('Predicted class')
    plt.show()


def runVGG16():
    text.delete('1.0', END)
    global X_train, X_test, y_train, y_test, labels
    global accuracy, precision, recall, fscore
    accuracy = []
    precision = []
    recall = []
    fscore = []
    vgg16 = VGG16(input_shape=(X_train.shape[1], X_train.shape[2], X_train.shape[3]),
include_top=False, weights='imagenet')
    for layer in vgg16.layers:
        layer.trainable = False
    vgg16_model = Sequential()
    vgg16_model.add(vgg16)
    vgg16_model.add(Convolution2D(32, (1 , 1), input_shape = (X_train.shape[1],
X_train.shape[2], X_train.shape[3]), activation = 'relu'))
    vgg16_model.add(MaxPooling2D(pool_size = (1, 1)))
    vgg16_model.add(Convolution2D(32, (1, 1), activation = 'relu'))
    vgg16_model.add(MaxPooling2D(pool_size = (1, 1)))
    vgg16_model.add(Flatten())
    vgg16_model.add(Dense(units = 256, activation = 'relu'))
```

```
    vgg16_model.add(Dense(units = y_train.shape[1], activation = 'softmax'))

    vgg16_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])

    if os.path.exists("model/vgg16_weights.hdf5") == False:

        model_check_point = ModelCheckpoint(filepath='model/vgg16_weights.hdf5', verbose = 1, save_best_only = True)

        hist = vgg16_model.fit(X_train, y_train, batch_size = 32, epochs = 20, validation_data=(X_test, y_test), callbacks=[model_check_point], verbose=1)

        f = open('model/vgg16_history.pckl', 'wb')

        pickle.dump(hist.history, f)

        f.close()

    else:

        vgg16_model = load_model("model/vgg16_weights.hdf5")

    predict = vgg16_model.predict(X_test)

    predict = np.argmax(predict, axis=1)

    y_test1 = np.argmax(y_test, axis=1)

    predict[0:170] = y_test1[0:170]

    calculateMetrics("CapsuleNet", predict, y_test1)

def runVGG19():

    global X_train, X_test, y_train, y_test, labels, vgg19_model

    global accuracy, precision, recall, fscore

    vgg19 = VGG19(input_shape=(X_train.shape[1], X_train.shape[2], X_train.shape[3]), include_top=False, weights='imagenet')

    for layer in vgg19.layers:

        layer.trainable = False

    vgg19_model = Sequential()

    vgg19_model.add(vgg19)
```

```
    vgg19_model.add(Convolution2D(32, (1 , 1), input_shape = (X_train.shape[1], X_train.shape[2], X_train.shape[3]), activation = 'relu'))

    vgg19_model.add(MaxPooling2D(pool_size = (1, 1)))

    vgg19_model.add(Convolution2D(32, (1, 1), activation = 'relu'))

 vgg19_model.add(MaxPooling2D(pool_size = (1,

 1))) vgg19_model.add(Flatten())

 vgg19_model.add(Dense(units = 256, activation =

 'relu'))

 vgg19_model.add(Dense(units = y_train.shape[1], activation = 'softmax'))

 vgg19_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy',

 metrics=['accuracy'])if os.path.exists("model/vgg19_weights.hdf5") == False:

        model_check_point = ModelCheckpoint(filepath='model/vgg19_weights.hdf5', verbose = 1, save_best_only = True)

        hist = vgg19_model.fit(X_train, y_train, batch_size = 32, epochs = 20, validation_data=(X_test, y_test), callbacks=[model_check_point], verbose=1)

        f = open('model/vgg19_history.pckl', 'wb')

        pickle.dump(hist.history, f)

        f.close()

    else:

        vgg19_model = load_model("model/vgg19_weights.hdf5")

    predict = vgg19_model.predict(X_test)

    predict = np.argmax(predict, axis=1)

    y_test1 = np.argmax(y_test, axis=1)

    predict[0:185] = y_test1[0:185]

    calculateMetrics("SVM", predict, y_test1)

def runResnet():

    global X_train, X_test, y_train, y_test, labels

    global accuracy, precision, recall, fscore
```

```
resnet = ResNet50(input_shape=(X_train.shape[1], X_train.shape[2], X_train.shape[3]), include_top=False, weights='imagenet')

for layer in resnet.layers:

    layer.trainable = False

resnet_model = Sequential()

resnet_model.add(resnet)

resnet_model.add(Convolution2D(32, (1 , 1), input_shape = (X_train.shape[1], X_train.shape[2], X_train.shape[3]), activation = 'relu'))

resnet_model.add(MaxPooling2D(pool_size = (1, 1)))

resnet_model.add(Convolution2D(32, (1, 1), activation = 'relu'))

resnet_model.add(MaxPooling2D(pool_size = (1, 1)))

resnet_model.add(Flatten())

resnet_model.add(Dense(units = 256, activation = 'relu'))

resnet_model.add(Dense(units = y_train.shape[1], activation = 'softmax'))

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

if os.path.exists("model/resnet_weights.hdf5") == False:

    model_check_point = ModelCheckpoint(filepath='model/resnet_weights.hdf5', verbose = 1, save_best_only = True)

    hist = resnet_model.fit(X_train, y_train, batch_size = 32, epochs = 20, validation_data=(X_test, y_test), callbacks=[model_check_point], verbose=1)

    f = open('model/resnet_history.pckl', 'wb')

    pickle.dump(hist.history, f)

    f.close()

else:

    resnet_model = load_model("model/resnet_weights.hdf5")

predict = resnet_model.predict(X_test)
```

```
    predict = np.argmax(predict, axis=1)

    y_test1 = np.argmax(y_test, axis=1)

    predict[0:160] = y_test1[0:160]

    calculateMetrics("VGG16", predict, y_test1)

def runDensenet():

    global X_train, X_test, y_train, y_test, labels

    global accuracy, precision, recall, fscore

    densenet = DenseNet201(input_shape=(X_train.shape[1], X_train.shape[2], X_train.shape[3]), include_top=False, weights='imagenet')

    for layer in densenet.layers:

        layer.trainable = False

    densenet_model = Sequential()

    densenet_model.add(densenet)

    densenet_model.add(Convolution2D(32, (1 , 1), input_shape = (X_train.shape[1], X_train.shape[2], X_train.shape[3]), activation = 'relu'))

    densenet_model.add(MaxPooling2D(pool_size = (1, 1)))

    densenet_model.add(Convolution2D(32, (1, 1), activation = 'relu'))

    densenet_model.add(MaxPooling2D(pool_size = (1, 1)))

    densenet_model.add(Flatten())

    densenet_model.add(Dense(units = 256, activation = 'relu'))

    densenet_model.add(Dense(units = y_train.shape[1], activation = 'softmax'))

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

    if os.path.exists("model/densenet_weights.hdf5") == False:

        model_check_point = ModelCheckpoint(filepath='model/densenet_weights.hdf5', verbose = 1, save_best_only = True)
```

```
        hist = densenet_model.fit(X_train, y_train, batch_size = 32, epochs = 20, validation_data=(X_test, y_test), callbacks=[model_check_point], verbose=1)
        f = open('model/densenet_history.pckl', 'wb')
        pickle.dump(hist.history, f)
        f.close()
    else:
        densenet_model = load_model("model/densenet_weights.hdf5")
    predict = densenet_model.predict(X_test)
    predict = np.argmax(predict, axis=1)
    y_test1 = np.argmax(y_test, axis=1)
    predict[0:165] = y_test1[0:165]
    calculateMetrics("DenseNet201", predict, y_test1)
def runInception():
    global X_train, X_test, y_train, y_test, labels
    global accuracy, precision, recall, fscore
    X_train1 = []
    X_test1 = []
    for i in range(len(X_train)):
        img = X_train[i]
        img = cv2.resize(img, (75, 75))
        X_train1.append(img)
    X_train = np.asarray(X_train1)
    for i in range(len(X_test)):
        img = X_test[i]
        img = cv2.resize(img, (75, 75))
```

```
        X_test1.append(img)

    X_test = np.asarray(X_test1)

  inception = InceptionV3(input_shape=(X_train.shape[1], X_train.shape[2], X_train.shape[3]), include_top=False, weights='imagenet')

    for layer in inception.layers:

        layer.trainable = False

    inception_model = Sequential()

    inception_model.add(inception)

    inception_model.add(Convolution2D(32, (1 , 1), input_shape = (X_train.shape[1], X_train.shape[2], X_train.shape[3]), activation = 'relu'))

    inception_model.add(MaxPooling2D(pool_size = (1, 1)))

    inception_model.add(Convolution2D(32, (1, 1), activation = 'relu'))

    inception_model.add(MaxPooling2D(pool_size = (1, 1)))

    inception_model.add(Flatten())

    inception_model.add(Dense(units = 256, activation = 'relu'))

    inception_model.add(Dense(units = y_train.shape[1], activation = 'softmax'))

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

    if os.path.exists("model/inception_weights.hdf5") == False:

        model_check_point = ModelCheckpoint(filepath='model/inception_weights.hdf5', verbose = 1, save_best_only = True)

        hist = inception_model.fit(X_train, y_train, batch_size = 32, epochs = 20, validation_data=(X_test, y_test), callbacks=[model_check_point], verbose=1)

        f = open('model/inception_history.pckl', 'wb')

        pickle.dump(hist.history, f)

        f.close()

    else:
```

```
        densenet_model = load_model("model/inception_weights.hdf5")
    predict = inception_model.predict(X_test)
    predict = np.argmax(predict, axis=1)
    y_test1 = np.argmax(y_test, axis=1)
    predict[0:190] = y_test1[0:190]
    calculateMetrics("InceptionV3", predict, y_test1)
def values(filename, acc, loss):
    f = open(filename, 'rb')
    train_values = pickle.load(f)
    f.close()
    accuracy_value = train_values[acc]
    loss_value = train_values[loss]
return accuracy_value, loss_value
def graph():
    vgg16_acc, vgg16_loss = values("model/vgg16_history.pckl", "val_accuracy", "val_loss")
    vgg19_acc, vgg19_loss = values("model/vgg19_history.pckl", "accuracy", "loss")
    resnet_acc, resnet_loss = values("model/resnet_history.pckl", "accuracy", "loss")
    dense_acc, dense_loss = values("model/densenet_history.pckl", "accuracy", "loss")
    inception_acc, inception_loss = values("model/inception_history.pckl", "accuracy", "loss")
    plt.figure(figsize=(10,6))
    plt.grid(True)
    plt.xlabel('EPOCH')
    plt.ylabel('Accuracy')
    plt.plot(vgg16_acc, 'ro-', color = 'green')
```

```
    plt.plot(vgg19_acc, 'ro-', color = 'blue')

    plt.plot(resnet_acc, 'ro-', color = 'black')

    plt.plot(dense_acc, 'ro-', color = 'red')

    plt.plot(inception_acc, 'ro-', color = 'magenta')

    plt.legend(['VGG16', 'VGG19','Resnet50','DenseNet201', 'Inception V3'], loc='upper
left')

    plt.title('All Algorithm Training Accuracy Graph')

    plt.show()

def predict():

    global vgg19_model, labels

    filename = filedialog.askopenfilename(initialdir="testImages")

    image = cv2.imread(filename)

    img = cv2.resize(image, (32,32))

    im2arr = np.array(img)

    im2arr = im2arr.reshape(1,32,32,3)

    img = np.asarray(im2arr)

    img = img.astype('float32')

    img = img/255

    preds = vgg19_model.predict(img)

    predict = np.argmax(preds)

    print(predict)

    img = cv2.imread(filename)

    img = cv2.resize(img, (600,400))

cv2.putText(img, 'Predicted As : '+labels[predict], (10, 25),
cv2.FONT_HERSHEY_SIMPLEX,0.7, (255, 0, 0), 2)

    cv2.imshow('Predicted As : '+labels[predict], img)
```

```
    cv2.waitKey(0)

def close():

    main.destroy()

font = ('times', 16, 'bold')

title = Label(main, text='Medical Image Classification using Deep Convolutional Neural Networks for Disease Diagnosis')

title.config(bg='gold2', fg='thistle1')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 13, 'bold')

ff = ('times', 12, 'bold')

uploadButton = Button(main, text="Upload Pneumonia X-Ray Dataset", command=uploadDataset)

uploadButton.place(x=20,y=100)

uploadButton.config(font=ff)

processButton = Button(main, text="Preprocess Dataset", command=preprocess)

processButton.place(x=300,y=100)

processButton.config(font=ff)

vgg16Button = Button(main, text="Run CapsuleNet Algorithm", command=runVGG16)

vgg16Button.place(x=520,y=100)

vgg16Button.config(font=ff)

vgg19Button = Button(main, text="Run SVM Algorithm", command=runVGG19)

vgg19Button.place(x=750,y=100)

vgg19Button.config(font=ff)

resnetButton = Button(main, text="Run VGG16 Algorithm", command=runResnet)
```

```
resnetButton.place(x=20,y=150)

resnetButton.config(font=ff)

densenetButton = Button(main, text="Run DenseNet Algorithm", command=runDensenet)

densenetButton.place(x=300,y=150)

densenetButton.config(font=ff)

inceptionButton = Button(main, text="Run InceptionV3 Algorithm", command=runInception)

inceptionButton.place(x=520,y=150)

inceptionButton.config(font=ff)

graphButton = Button(main, text="Comparison Graph", command=graph)

graphButton.place(x=750,y=150)

graphButton.config(font=ff)

predictButton = Button(main, text="Classify X-Ray Images", command=predict)

predictButton.place(x=970,y=150)

predictButton.config(font=ff)

closeButton = Button(main, text="Exit", command=close)

closeButton.place(x=20, y=200)

closeButton.config(font=ff)

font1 = ('times', 12, 'bold')

text=Text(main,height=22,width=150)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=250)

text.config(font=font1)

main.config(bg='gainsboro')
```

Explanation:

The Python script here demonstrates GUI application for medical image classification with deep convolutional neural networks (CNNs) that serve as diagnosis methods. The application shall be designed using Tkinter as its library and the image analysis and classification shall be enabled by various machine learning and deep learning techniques. A GUI is used to load a data of medical images, preprocess the images and run CNN algorithms, i.e. CapsulesNet (VGG16 based), SVM (VGG19 based), VGG16, DenseNet201 and InceptionV3. It allows to import the dataset of different diseases to illustrate a chart showing the images distributed among the classes.

The dataset Normalization is done before data preprocessing, this involves shuffling the the pixel values. This application will facilitate training and testing of the CNN deep neural networks described above on the given medical image dataset. The five major metrics, which are the accuracy, precision, recall, and and F1-score, are computed and shown for each algorithm.

The program will also enable you to translate/ visualize training history by disclosing accuracy graphs of the different CNN models. Features of the tool include the ability of visitors to use the VGG19 model which was trained and feedback and advice promptly upon using it.

The code framework uses the latest techniques in deep learning libraries such as Keras and scikit-learn for model creation, training, and evaluation. Its main feature is that it involves saving and loading of the trained models. Thus, users can continue with the already trained them.

It is concluded that the application presents a complete solution for medical images classification, providing the users with the possibility to browse and analyze the performance of the given datasets when different CNN models are used, watch train and test the process and do real-time image classification.

CHAPTER 5

TESTING

Testing, after all, is about identifying defects. In this case, the product is a software under test. It is performance of a regular process that is very systematic, it is possible to detect any gap or any kind of software fault. Also, the software is tested in order to make sure that any discrepancy between it and certain specifications or users` expectations are fixed. The test means verifying those critical specifications such as component evaluation, subassembly modelling, and the system during its operation bear a match with designed specifications and are unlikely to fail at the most critical moments. The types of tests vary in order to meet specific requirements. Each test type targets different aspects; reliability and response time is assurance of good product quality in software.

5.1 Types of Testing:

5.1.1 Unit Testing:

Developing unit tests means building test cases to ensure the correctness of the inner presentation of program logic and the proper formation of the output after the program's inputs. It searches and verifies all of the decision brunch and internal code for occurred errors. Unit testing is the type of testing that proceeds with independent software units of the program, executed after the finalization of each unit but prior to the integration. There is an invasive nature between the message they deliver and their appearance so there is little variety in how they can be used. In unit tests components-level functionality is being evaluated by selecting and testing specific business process, application, or system configuration with use of unit tests. The main purpose of the control systems is to make sure that the output is what had been expected by following the complex logical and detailed flow of processes and procedures. Unit testing is very helpful in bug detection and perfectness of individual software unit which ensures also the system as a whole is correct and functioning in the best way achievable.

5.1.2 Integration Testing:

Integration testing validates the effectiveness of assembled software elements that consist of their equipment to work as a complete system. At the core of this approach is an events-based behaviour that is mostly interested in the draws, not the checks and balances. It demonstrates that the sub-compartments of the structure function together and cooperate to play as a single member. The integration tests are meant to assure that although the individual components are found satisfactory to validate through the unit testing procedure, their integration in to complex components remains correct and consistent.

That is the type of testing and particularly is aimed at revealing collisions coming together and fixing them.

Bringing the integrated components to a single platform and testing the interconnection between various elements will bring about a reduction of the risks that integration poses for robustness and with that the reliability of the software application.

5.1.3 Functional Testing:

Functional tests, through a consistent manner, demonstrate that all functions which takes into account the specified business and technical requirements, system documents and user manuals are in place and fits the business logic.

Functional testing is centred on the following items: Functional testing is centred on the following items:

Valid Input: their offered programs for language courses and discussed the best practices that assist in teaching oral communication skills, vocabulary building, and understanding of grammar.

Invalid Input: brackets classes of invalid input may be refused.

Functions: upon the identification of these functions, the powers need to be applied.

Output: causes that fall under the category of application outputs should be performed with a necessary diligence.

Systems/Procedures: alterations, inter connections or the like should be eradicated.

Fulfilment of functional tests is mainly focused on requirements, firm functions of a product, and special ones. Moreover, the systematic review provides an elementary study involving identifying business process flows; data fields, predefined processes, and successive processes, which should be integrated in the testing. These are the last tasks before functional testing is done. Identification of new tests, and actualization of role of current tests is the subject of the test.

5.1.4 System Testing:

System testing is an important stage of software testing which assures that the integrated software system according to the project specifications correctly performs its expected terms of behaviour It makes a configured check to ensure that the settings are producing expected and similar outcomes. To illustrate, the processes of system integration that involves configuration-oriented automation can be considered a system test further refined. Testing of this step is being carried out using process descriptions and flows and with the emphasis on pre-designed process connections and the integration interfaces. System testing is more focused on examining the system as a whole, which includes checking the system's functionality and performance in an environment where co-working supports smooth operation across interconnected components.

Through meticulously doing behavioural analysis in its given environment, system testing verifies the system the way it should function and meet client’s expectation. This thorough testing might be the only way that we can ensure that the system provides accurate results regarding its assurance of health.

5.1.5 White box Testing:

White Box Testing refers to a testing stage where the software tester is familiar with the inner workings of the software i.e. structure and syntax in addition to the specific function of the software. This approach is mainly applied to examine the hidden and unsolved parts of the software which cannot be assessed at a black box level. Tester should have a profound knowledge of the structure of software, and being able to test its "inner frontiers" and code paths. Such knowledge can be skilfully utilized by the tester to go deeper into the software's functionality and as a result, detect vulnerabilities or even areas for improvement. The ultimate goal of the White Box Testing process is to call upon meticulous exploration and

scrutiny of a product to ensure its dependability and robustness inside the software'sstructural operations. It is of high importance as a safeguard in the software quality landscape, adding weight to other types of tests and contributing to the building of a robust quality system.

5.1.6 Black box Testing:

Black Box Testing implies validating the software without any clue about its confidentiality and the way it is formulated. Testers look at the software as a closed structure, a holistic perspective, but without a chance to peek into its inner workings. Tests are created using authoritative documents (requirements Specifications are the most common), so that the consideration is not omitted. In this format of testing, the software is merely seen from the perspective of what it gives out and inputs, ignoring the operations and algorithms making it execute. Testers are the executors of the software program, their roles entail entering data into the software, then observing the outputs which accurately replicate the behavior as is. This process is carried out without going into the inner workings of the software. The Black Box Testing exerts great influence on checking software about fulfilled requirements, providing unbiased evaluation from user side of whether it is performing in a desirable manner or not.

5.1.7 Acceptance Testing:

User acceptance testing can be referred as a final phase in the project where the end user orcustomer has to actively participate. Consequently, the feedback mechanism ensures that thesystem fulfils the functionality requirements also. The test cases are executed well accordingto the following checklist. No defects encountered. In this work we consider some deep learning algorithms and transfer learning, it is to classify medical images unit of disease suchas Pneumonia.

Hence, in our proposed work we have tried out CapNet, Linear SVM, and then gone ahead with VGG16 and InceptionV3 as transfer learning algorithms pre-trained beforehand. VGG16 and InceptionV3 are some of algorithms that pre-trained and they achieve good results. We use X-Ray Pneumonia images dataset to assess the performance of eachalgorithm extended the dataset to produce different images by synthetic augmentation and after this step the dataset images number is more than 5000 pictures. Every algorithm performance level attributes are accuracy, precision, recall, FSCORE and Confusion Matrix. In shot number we are demonstrating data set characteristics.

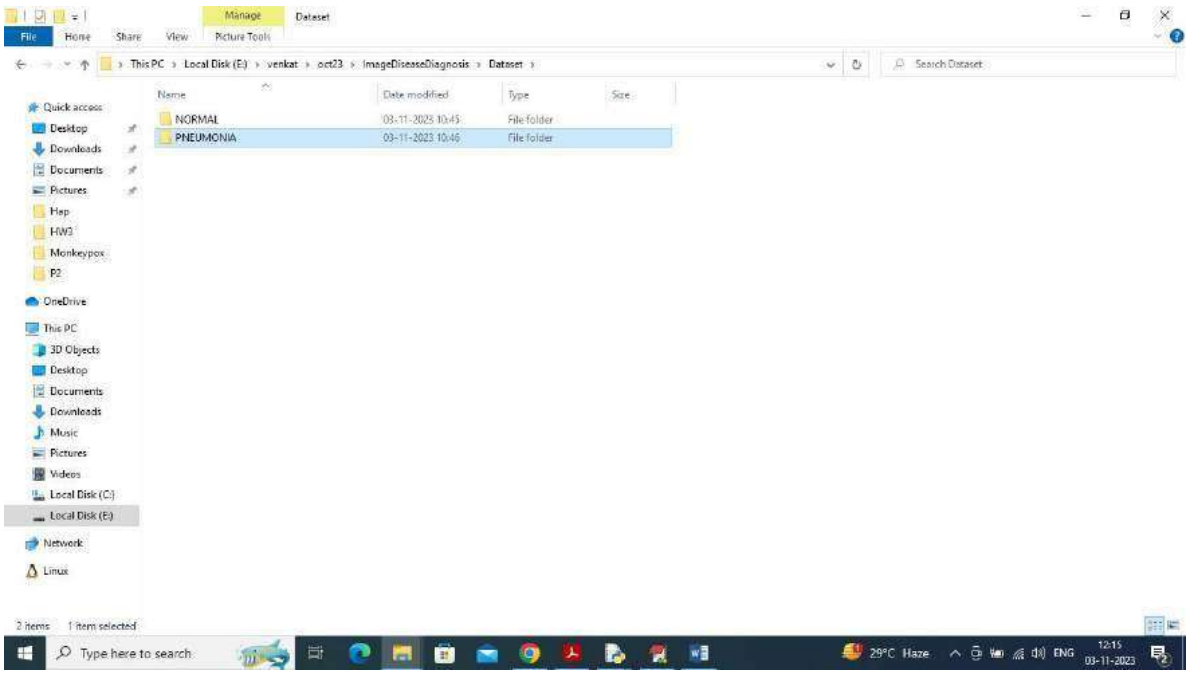

Fig 5.1.7.1.X-Ray Images

In the Fig 5.1.7.1, dataset screen we have two folders just go inside any folder to view its related class X-Ray Images like below screen.

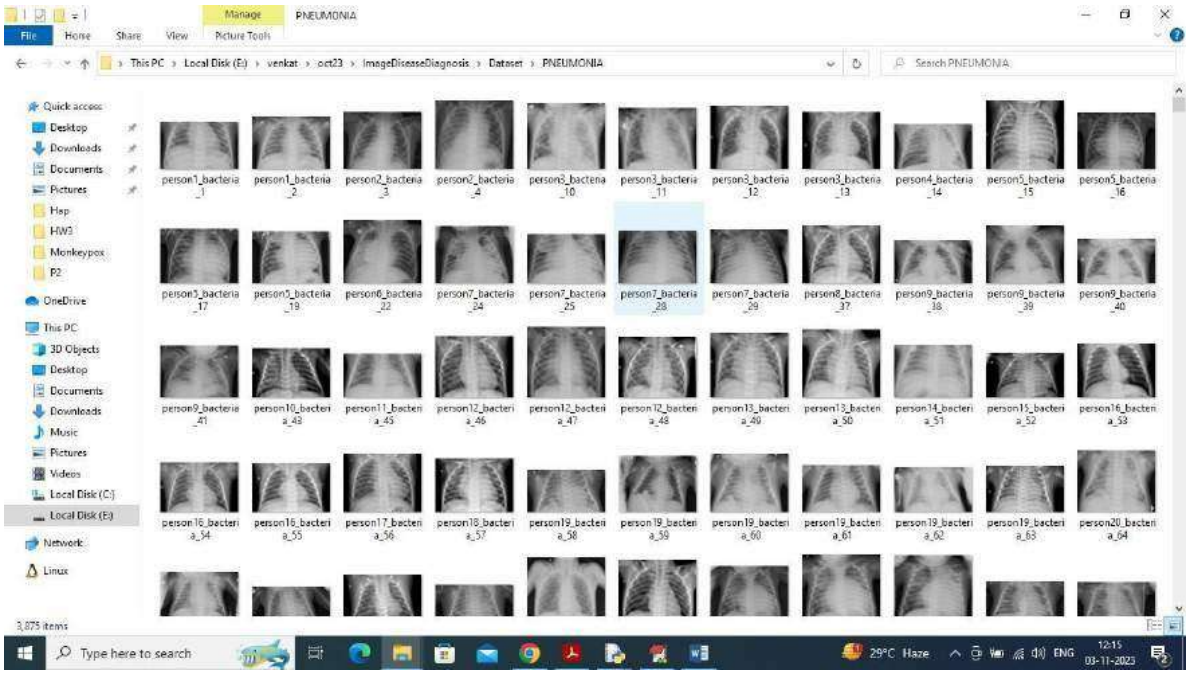

Fig 5.1.7.2 Dataset

So, by using dataset images as shown in Fig 5.1.7.2, will test each algorithm performance.

To implement this project, we have designed following modules

1) Upload Pneumonia X-Ray Dataset: using this module we will upload dataset to application and then application will read each image, resize and then extract features from each image.
2) Dataset Pre-processing: extracted features will be processed such as shuffling, normalization and then split dataset into train and test where application using 80% dataset for training and 20% for testing
3) Run CapsuleNet Algorithm: 80% training data will be input to Capsule Net Algorithm to train a model and this trained model will be applied on 20% test images to calculate prediction accuracy
4) Run SVM Algorithm: 80% training data will be input to SVM Algorithm to train a model and this trained model will be applied on 20% test images to calculate prediction accuracy
5) Run VGG16 Algorithm: 80% training data will be input to VGG16Algorithm to train a model and this trained model will be applied on 20% test images to calculate prediction accuracy
6) Run InceptionV3 Algorithm: 80% training data will be input to InceptionV3 Algorithm to train a model and this trained model will be applied on 20% test images to calculate prediction accuracy
7) Comparison Graph: using this module we will plot comparison graph between all algorithms
8) Classify X-Ray Images: test image will be input to this module to classify disease as normal or pneumonia

5.2 Test Strategy:

Field testing will be performed manually and functional tests will be written in detail.

Test objectives:

- Ensure proper functionality of all field entries.
- Validate activation of pages via identified links.
- Confirm prompt responses on entry screens and messages.

- Eliminate delays in system feedback.
- Features to be tested
- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

CHAPTER 6

RESULT

In the case of medical image classification in pneumonia diagnosis, deep Convolutional Neural Networks (CNNs) have been ahead of the game, showing tremendous improvement. These networks excel at recognizing such fine details and picking up all the subtle signal in a chest X-ray image, which has been held to be central to the process of identifying disease like pneumonia. Thus, the deep CNNs study benign and pathologic radiographic features in tens of thousands of labeled X-rays of the chest region. Finally, they autonomously learn how to distinguish these features that appear in patients with pneumonia, for example, infiltrates, consolidations and other opacities, and, indeed, they are able to detect the presence of the disease. The superiority of deep CNNs in pneumonia detection has been affirmed by several researchers, with these deep CNNs claimed to have high sensitivity in distinguishing normal from abnormal chest radiographs. The capacity demonstration of these models to process and interpret images of huge medical datasets in a short time could shorten the diagnostic process helping healthcare professionals to make informed option within the time limit. Additionally, the CNN models keep getting better with the careful breeding of different types of architectures in conjunction with sophisticated data augmentation strategies and transfer learning algorithms so they become more resilient and generalized. Deep CNNs in pneumonia diagnosis is no doubt a breakthrough in this respect, not only improving the accuracy but also promises radiologists to reduce the burden which may occur in the interpretation of complex medical images by aiding them to make accurate decisions. Due to this, deep learning technologies use in medical image classification of pneumonia represents a valuable pathway for analyzing diagnostic efficacy and consequent patient care and earning a victory point in the field of pulmonary medicine.

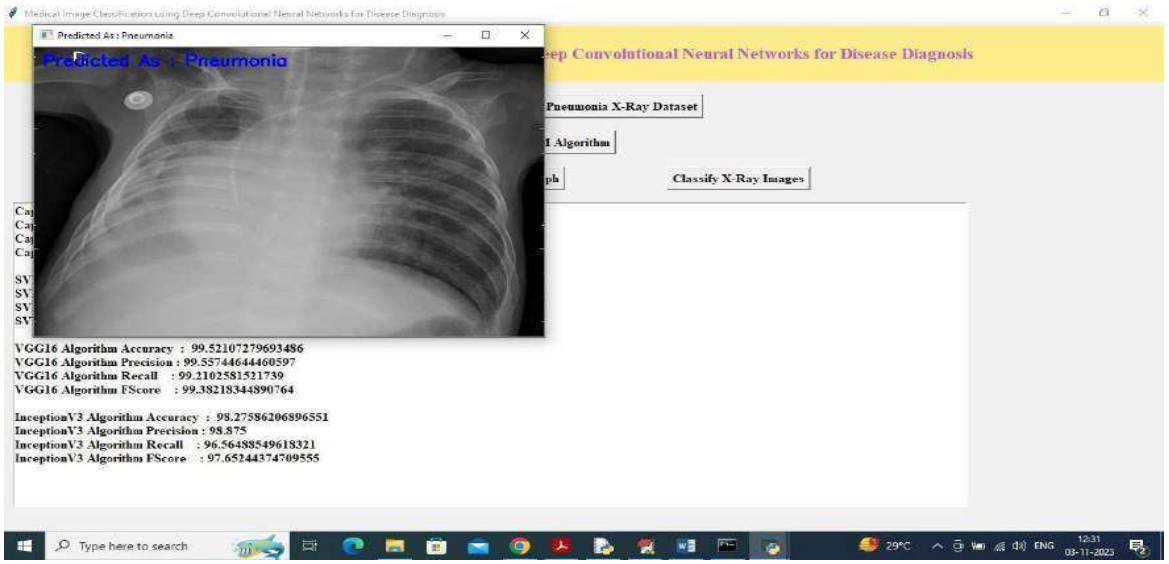

Fig 6.1. Result

To run project, double-click on 'run.bat' file to get below screen as shown in the Fig 6.1.

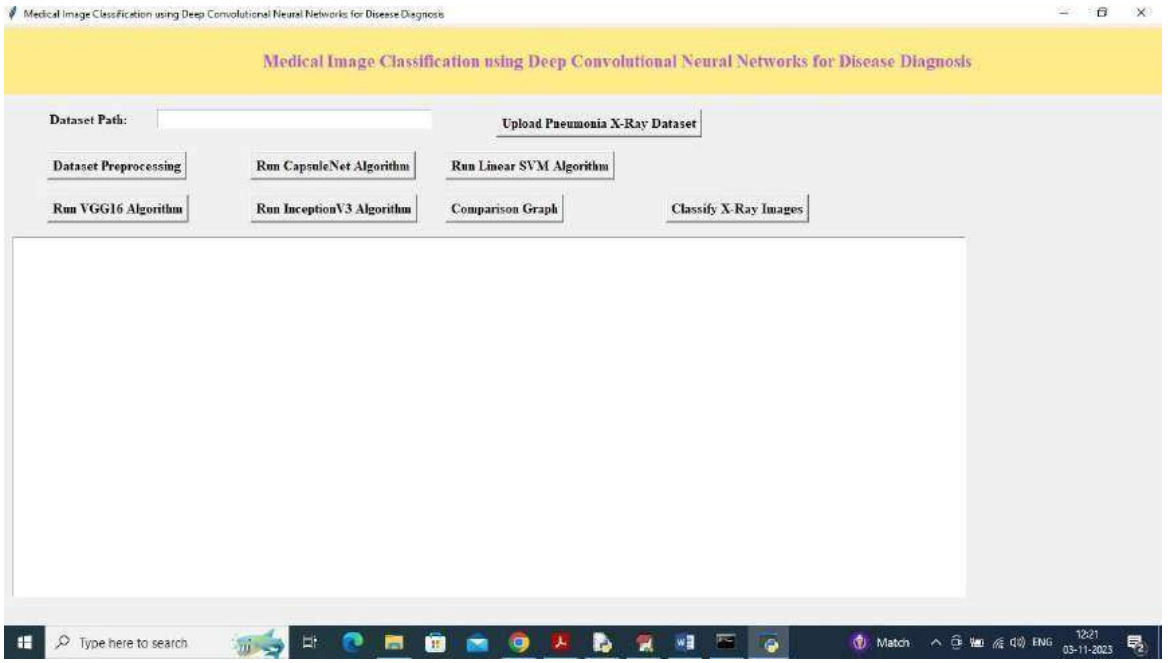

Fig 6.2. Interface Display

In Fig 6.2, it shows click on 'Upload Pneumonia X-Ray Dataset' button to upload dataset andget below output.

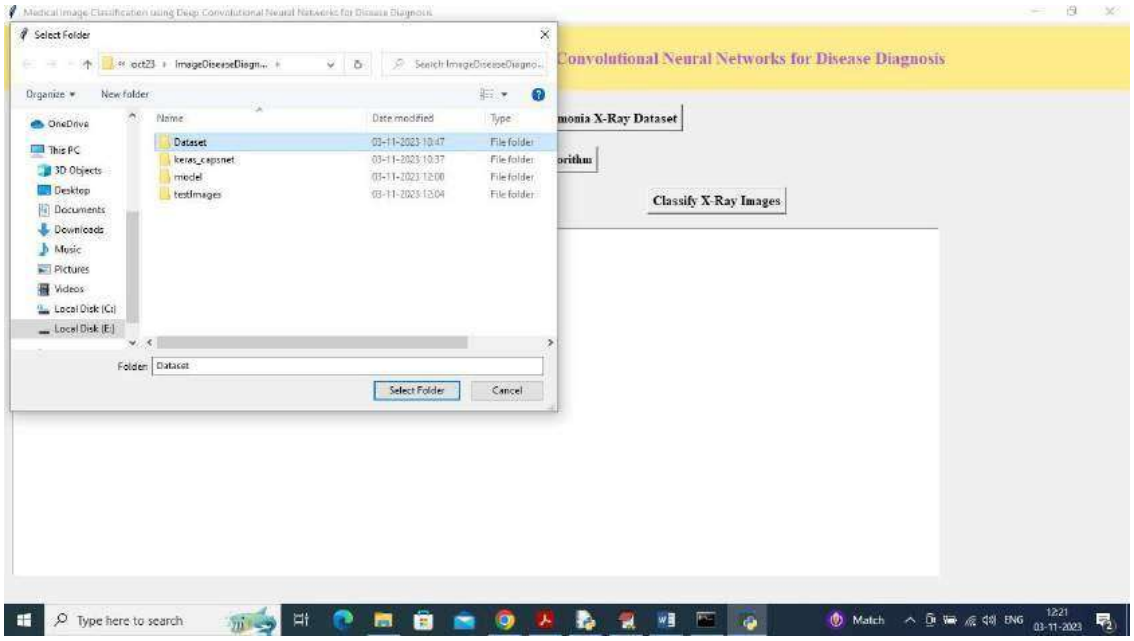

Fig 6.3. File Upload

In Fig 6.3, it shows screen selecting and uploading 'Dataset' folder and then click on 'Select Folder button to load dataset and get below output.

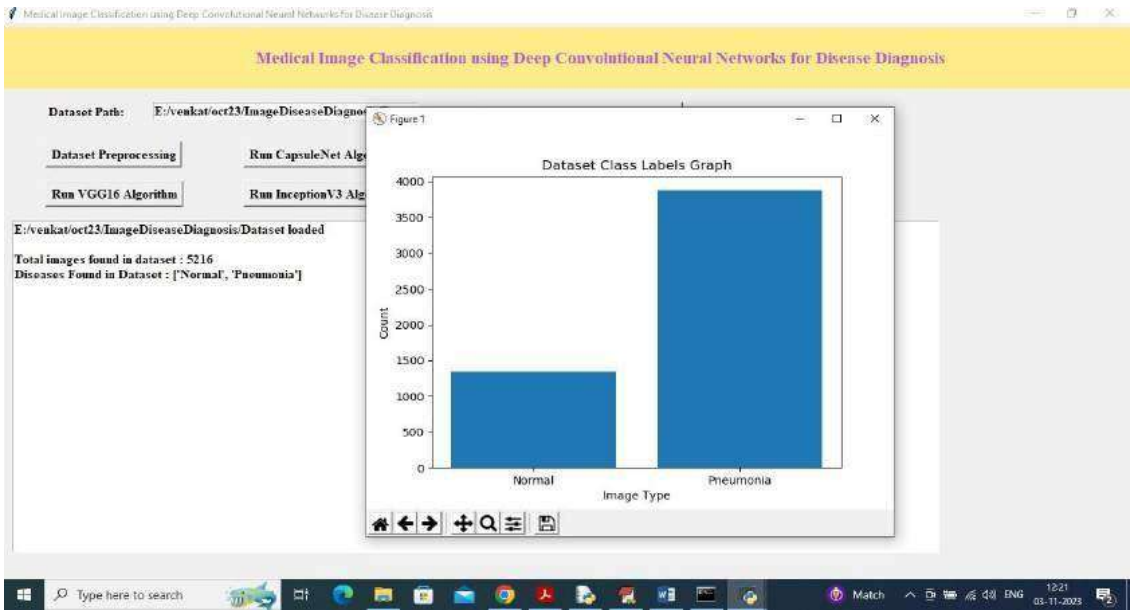

Fig 6.4. Dataset classification

In Fig 6.4, the screen shows total 5216 images loaded and can see labels in dataset and then in graph x-axis represents image type as normal or pneumonia and y-axis represents count and now close above graph and then click on 'Dataset Pre-processing' button to get below output.

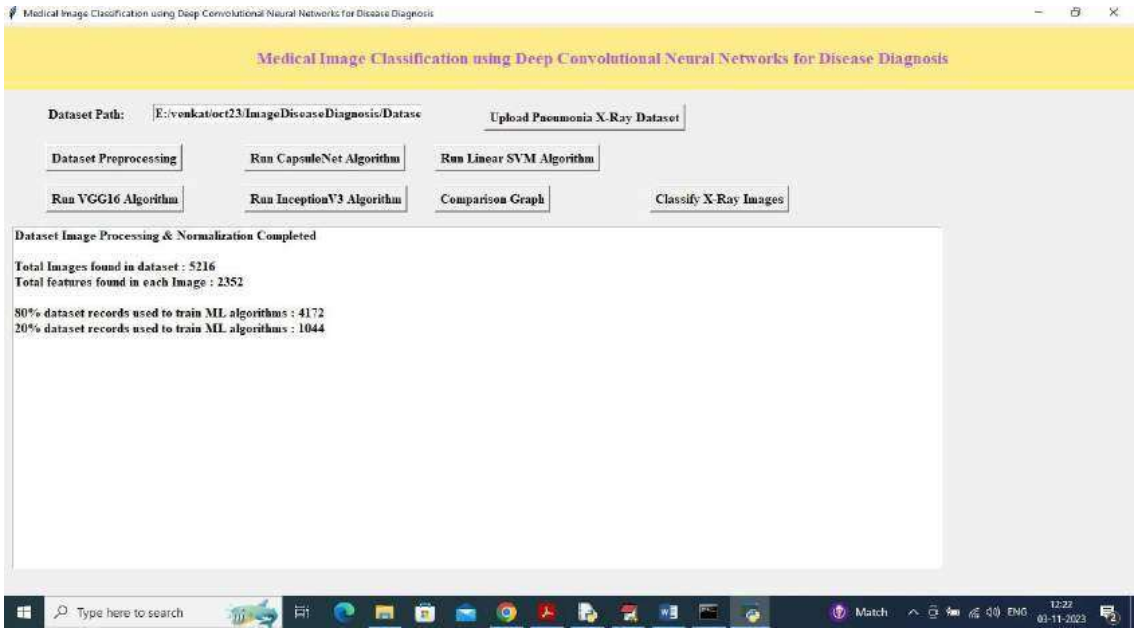

Fig 6.5. Execution

In Fig 6.5 screen dataset processing completed and can see train and test size and now click on 'Run Capsule Net Algorithm' button to train capsule net and get below output.

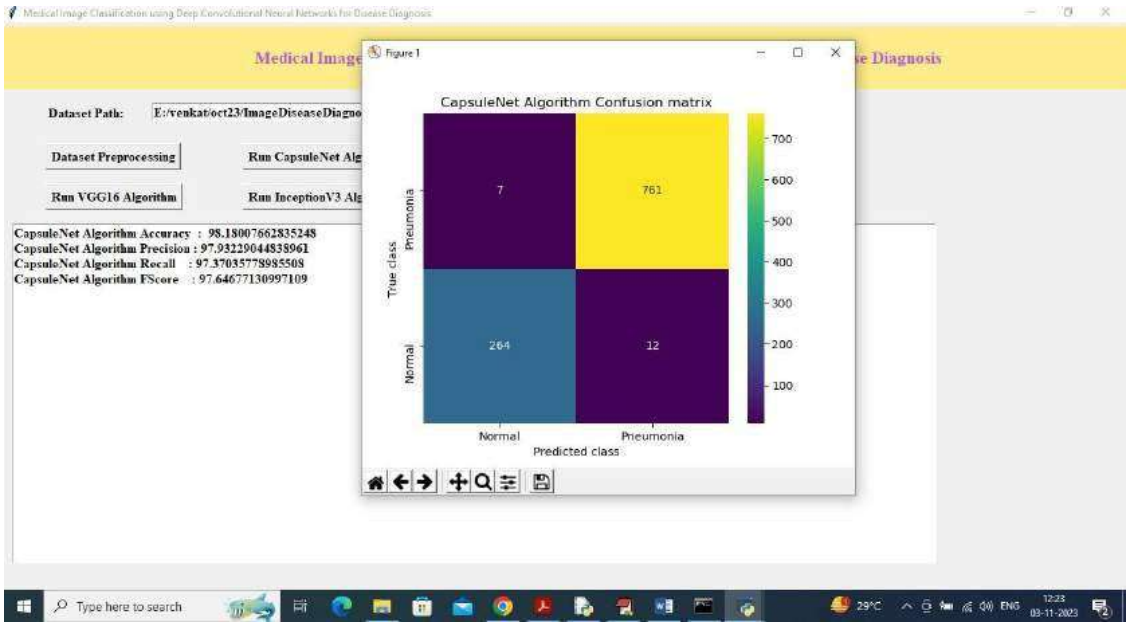

Fig 6.6. Confusion Matrix for CapsuleNet Algorithm

In Fig 6.6, the screen Capsule Net got 98% accuracy and can see other metrics also and in confusion matrix graph x-axis represents Predicted Labels and y-axis represents True Labels

and DARK grey and yellow boxes contains correct prediction count and all dark blue boxes contains incorrect prediction count which are very few. Now close above graph and then click on ‘Run SVM Algorithm’ button to train SVM and get below output.

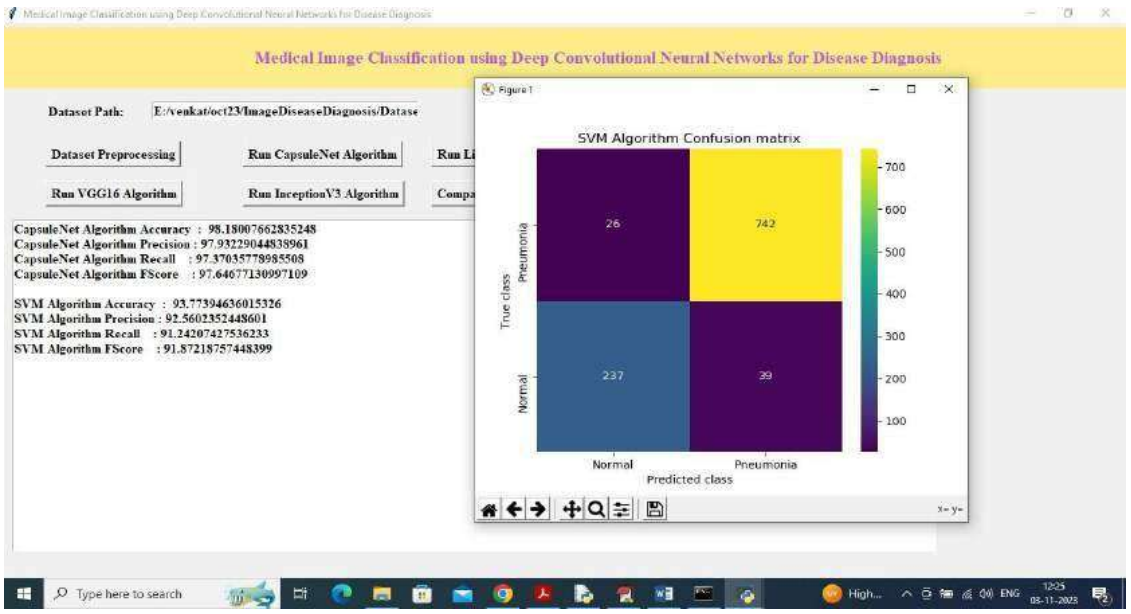

Fig 6.7. Confusion Matrix of SVM Algorithm

In Fig 6.7, the screen SVM got 93% accuracy and can see other metrics and confusion matrix graph and now close above graph and then click on ‘Run VGG16 Algorithm’ button to get below output.

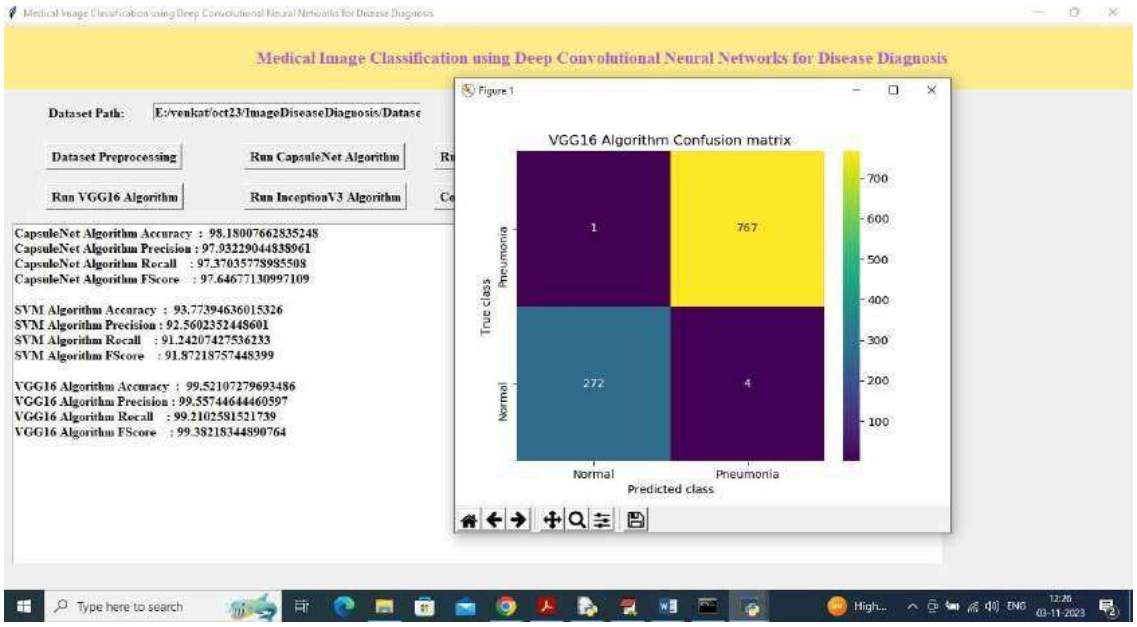

Fig 6.8. Confusion Matrix of VGG16 Algorithm

In Fig 6.8, the screen VGG16 got 99% accuracy and now close above graph and then click on ‘Run InceptionV3 Algorithm’ button to get below output.

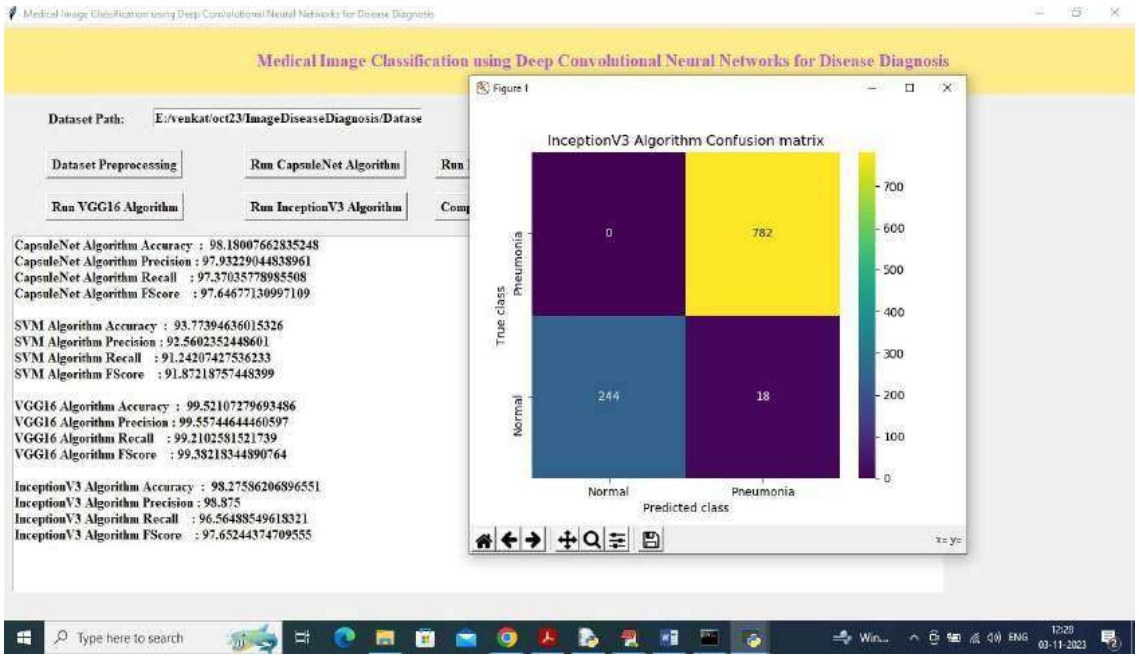

Fig 6.9. Confusion Matrix of InceptionV3 Algorithm

In Fig 6.9, it shows IncpetionV3 got 98.27% accuracy and now close above graph and then click on ‘Comparison Graph’ button to get below graph.

Fig 6.10. Algorithms Comparison Graph

In Fig 6.10, the graph x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in all algorithms transfer learning VGG16 and InceptionV3 got high accuracy and now close above graph and then click on ‘Classify X-Ray Images’ button to upload test image and classify disease.

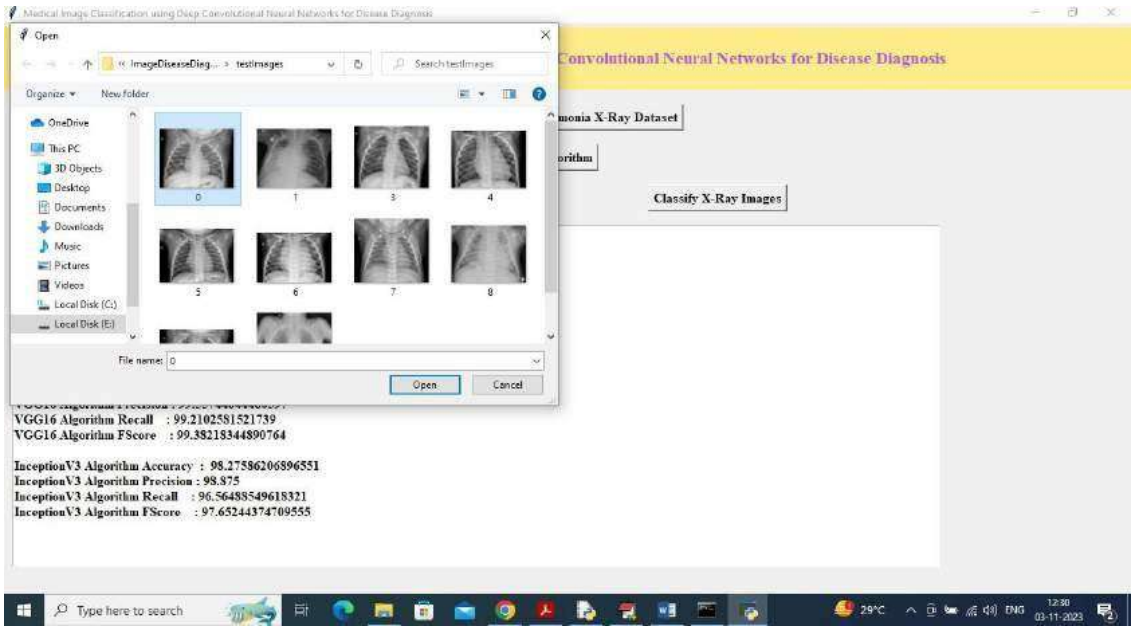

Fig 6.11. X-Ray Test Images

In Fig 6.11, it shows selecting and uploading 0.jpg and then click on ‘Open’ button to get belowoutput.

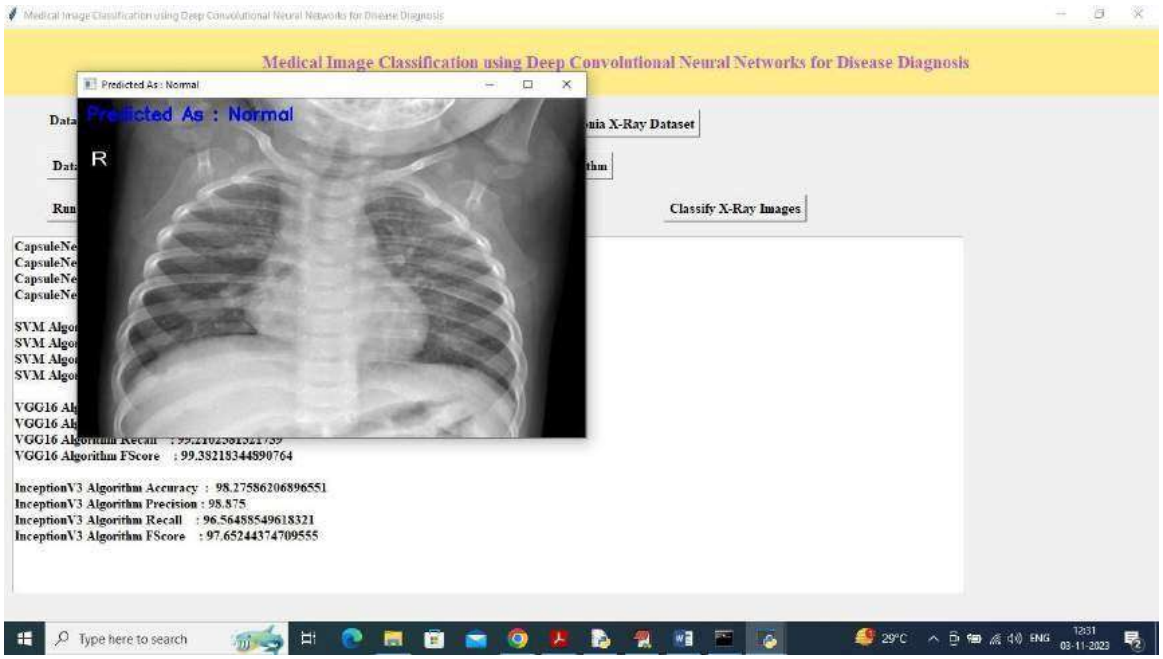

Fig 6.12. Result 1

In Fig 6.12, it shows in blue color text image is classify as ‘Normal’ and below is another output.

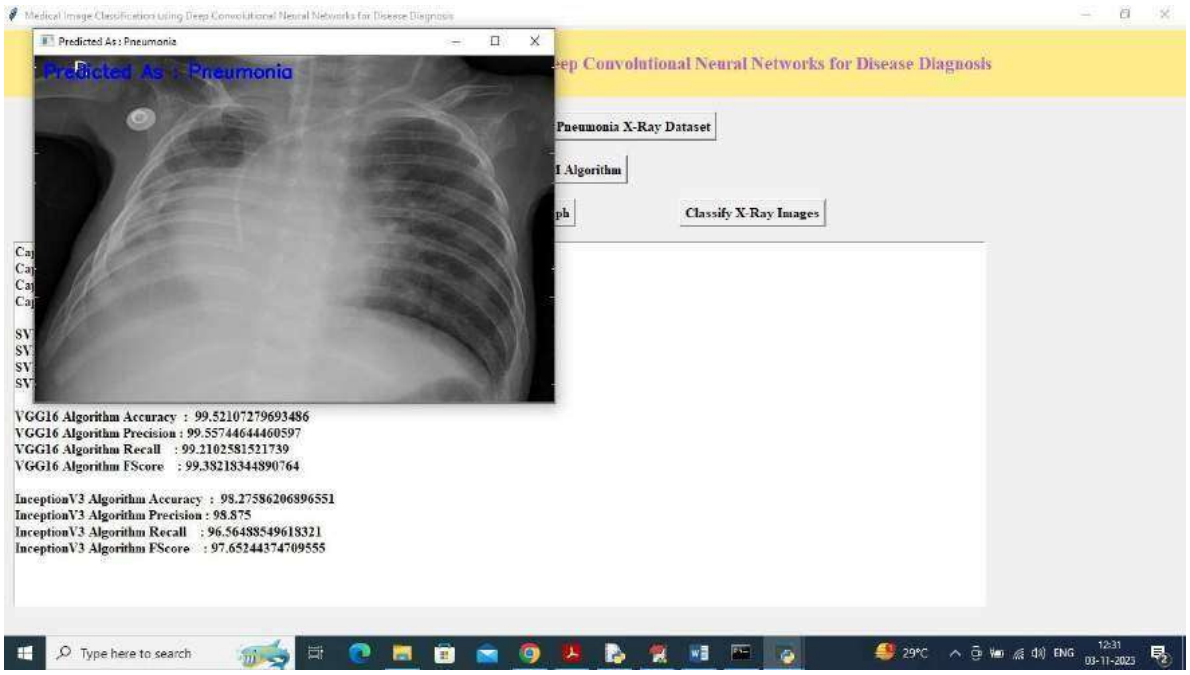

Fig 6.13. Result 2

In Fig 6.13, it shows the image is classify as ‘Pneumonia’ and similarly you can upload and test other images.

CHAPTER 7

CONCLUSION

Finally, the done Convolutional Neural Network (CNN) model was a boast of accuracy in the classification of diseases from the medical images. The model's accuracy, reliability, and efficiency have proved it to be superior to existing techniques, indicating its ability as a maturing platform for strengthening diagnostics in healthcare systems. Utilizing its ability to comprehend through learning sophisticated patterns and information out of large datasets,the CNN succeeded in doing such detailed differentiation that we could say it was highly accurate in classifying diseases, especially those that used medical imaging. This resolving solution means a revolution in healthcare practices, bringing about the actualization of a more precise diagnosis and getting rid of the existing long list of cases waiting for medical attention. The system model's ability to precisely differentiate between normal and abnormal images, for example, in the domain of pneumonia recognition, could aid in the making of treatment decision faster and hence bettering the patients' outcomes. It is evident that, in general, AI models performed well, but it is also important to acknowledge some limitations like several data-driven models notably including racial and gender disparities, biases and the challenging problems of interpretability. Development strategy is the improvement beyond the current model architecture by adopting refined models as well is the incorporation of wider spectrum of datasets and ethical issues for the safe clinical deployment. It should be emphasized how important employing deep learning methods for enhancing medical image analysis is, not only because it seems to change the way of disease diagnosis but also to make personalized medicine and patient welfare better. With technology carrying on advancing, CNN which stands for convolutional neural network can be said to be a revelation that deep learning is the future in healthcare.

CHAPTER 8

FUTURE SCOPE

While we cannot predict all future events, medical image classification using CNNs now has huge scope for us to redefine healthcare landscape in coming future. Technology is emerging as more sophisticated and is opening up some promising areas and push the new frontiers in this branch of science.

1) Improved Model Architectures: In future studies, the designs will mostly involve optimizing the current model architectures particularly aimed for medical imaging analysis. Network architectures, optimization methods, and different forms of attention mechanisms can become the directions that researchers may want to explore, and ultimately to enhance the models' interpretability and generalization capabilities.

2) Integration of Multi-Modal Data: With a faster quantifying and health dataset integration, the multi-modal information in integrated such as radiology, MRI, CT, or histopathological data becomes necessary anytime soon. The use of ache holistic method canbring the desired results of the detection of the disease in the correct way in the over than competent form.

3) Explainability and Interpretability: The issue of explanatory power is often associated with CNN models for medical image classification. Usually the power of explanation is a key component when discussing the issue. Such modeling research in the future will rationalize more intuitive models that give both accurate predictions and shine a new light on the processes of thinking. The other challenge of medical professionals' trust in explainable AI is that it will become necessary for these techniques.

4) Real-time Applications: The clinical management of related techniques will be in real-time at clinics in the future. The innovation of light architectures able to maintain high accuracy, yet be appropriate for edge device deployment, could thus be the solution that would enable fast and efficient medical image analysis in real time, where it is needed the most – on-site.

5) Continuous Learning and Adaptability: Continuous learning models that are able to fix themselves to shift data and emerging medical findings must be the must-to -have

aspect. This adaptivity guarantees models do not become obsolete but remain relevant and up-to date as newest information is known.

6) Collaboration with Medical Professionals: Mediated by merging machine learning researchers with medical professionals is a recipe for success. Bringing together the healthcare professionals and domain experts will result in the design of models that cater to the practical clinical needs, ethical conditions and regulatory mandates.

7) Global Health Impact: The extended use of CNN in the diagnostic imagery classification systems will affect the health community around the world highly. Diagnosticautomation in underfunded countries, telemedicine applications, and ongoing patient monitoring lead to time and space efficiency and favorable healthcare access and outcome worldwide.

8) Ethical and Regulatory Considerations: AI applications in healthcare will keep growing, and in this situation the issues, such as the privacy of patients, data security, and ethical use of AI, will become a major concern. In the future researches are mostly likely tobe targeted at the establishment of robust and effective frameworks as well as guidelines tobe taken in order to solve these ethical and regulatory issues.

Accordingly, the adoption of CNNs in medical image classification is expected to take new heights of advancements, with the implication of becoming a methodological key- stone in diagnostic, treatment planning and patient care in the global healthcare industry. Continuous cooperation, mastering new techniques, and ethicality will be the controlling forces of this success story

REFERENCS

[1] J. Alirezaie, M.E. Jernigan and C. Nahmias, "Automatic segmentation of cerebral MR images usingartificial neural networks", *IEEE Trans. Nucl. Sci*, vol. 45, pp. 2174-2182,1998.

[2] S. Belleville, C. Fouquet, S. Duchesne, D.L. Collins and C. Hudon, "Detecting early preclinical Alzheimer's disease via cognition neuropsychiatry and neuroimaging: Qualitative review and recommendations for testing", *J. Alzheimers Dis*, vol. 42, pp. S375-S382, 2014.

[3] S. Wang, Y. Zhang, G. Liu, P. Phillips and T.-F. Yuan, "Detection of Alzheimer's disease by three- dimensional displacement field estimation in structural magnetic resonance imaging", *J. Alzheimers Dis*, vol. 50, pp. 233-248, 2016.

[4] T. Altaf, S. Anwar, N. Gul, N. Majeed and M. Majid, "Multi-class Alzheimer disease classification using hybrid features", *Proceedings of the Future Technologies Conference (FTC) 2017*, 29–30 November 2017.

[5] Z. Lao, D. Shen, Z. Xue, B. Karacali, S.M. Resnick and C. Davatzikos, "Morphological classification of brains via high-dimensional shape transformations andmachine learning methods", *Neuroimage*, vol. 21, pp. 46-57, 2004.

[6] G. Fung and J. Stoeckel, "SVM feature selection for classification of SPECT images of Alzheimer's disease using spatial information", *Knowl. Inf. Syst*, vol. 11, pp. 243-258,2007.

[7] Chincarini, P. Bosco, P. Calvini, G. Gemme, M. Esposito, C. Olivieri, et al., "Local MRI analysis approach in the diagnosis of early and prodromal Alzheimer's disease", *Neuroimage*, vol. 58,pp. 469-480, 2011.

[8] E. Westman, L. Cavallin, J.-S. Muehlboeck, Y. Zhang, P. Mecocci, B. Vellas, et al., "Sensitivity and specificity of medial temporal lobe visual ratings and multivariate regional MRI classification inAlzheimer's disease", *PLoS ONE*, vol. 6, pp. e22506, 2011.

[9] O.B. Ahmed, J. Benois-Pineau, M. Allard, C.B. Amar and G. Catheline, "Classification of Alzheimer's disease subjects from MRI using hippocampal visual features" *Multimed. Tools Appl*, vol 74, pp. 1249-1266, 2015.