BRAIN TUMOR DETECTION USING CNN

Major Project submitted in partial fulfillment of the requirement for the award of the Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Under the esteemed guidance of

Mr.S.RAMANJANEYULU

Assistant professor

By:

V.Naga Tejaswi 15R11A05H3

Department of Computer Science & Engineering

Accredited by NBA

Geethanjali College of Engineering & Technology

(UGC Autonomous)

(Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi)

Geethanjali College of Engineering & Technology

(UGC Autonomous)

(Affiliated to JNTUH, Approved by AICTE, New Delhi)

Cheeryal (V), Keesara (M), R.R.Dist.-501 301.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Accredited by NBA

CERTIFICATE

This is to certify that the B.Tech Mini Project report entitled "**BRAIN TUMOR DETECTION USING CNN"** is a bonafide work done by **V.Naga Tejaswi (15R11AO5H3),** in partial fulfillment of the requirement of the award for the degree of Bachelor of Technology in "**Computer Science and Engineering**" from Jawaharlal Nehru Technological University, Hyderabad during the year 2015-2019.

Internal Guide
Mr.S.Ramanjaneyulu
(M.Tech)
Assistant professor

HOD - CSE
Dr D.S.R Murthy
(M.Tech,Phd)

External Examiner

Geethanjali College of Engineering & Technology

(UGC Autonomous)

(Affiliated to JNTUH Approved by AICTE,New Delhi)

Cheeryal (V), Keesara (M), R.R.Dist.-501 301.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Accredited by NBA

DECLARATION BY THE CANDIDATE

I **Naga Tejaswi**bearing Roll No. **15R11A05H3**hereby declare that the project report entitled **"BRAIN TUMOR DETECTION USING CNN"** is done under the guidance of **Mr.S.Ramanjaneyulu**, **Assistant professor**, Department of Computer Science and Engineering, Geethanjali College of Engineering and Technology, is submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering**.

V.Naga Tejaswi -15R11A05H3

Department of CSE,

Geethanjali College of Engineering and Technology,

Cheeryal.

ACKNOWLEDGEMENT

We are greatly indebted to the authorities of Geethanjali College of Engineering and Technology Cheeryal, Medchal Dist. For providing us the necessary facilities to successfully out this mini project work titled "**BRAIN TUMOR DETECTION USING CNN".**

Firstly, we thank and express us solicit gratitude to **Mr.S.Ramanjaneyulu Assistant professor,** Internal guide .Geethanjali College of Engineering and Technology, for this invaluable help and support which helped us a lot in successfully completing our mini project.

Secondly, we express our gratitude to Dr D.S.R.Murthy**, Head of Department, CSE** for being moral support throughout the period of study in **GCET**.

We would like to express our sincere gratitude to our principle **Dr.Udaya kumar susarla** for providing the necessary infrastructure to complete our project.

Finally, we would like to express our heartfelt thanks to our parents who were very supportive both financially and mentally and for their encouragement to achieve our set goals.

ABSTRACT

Brain tumor detection and classification is the most difficult and vapid task in the area of image preparing (in Medical Diagnosis). MRI (Magnetic Resonance Imaging) is a medicinal procedure, generally preferred by the radiologist for representation of inner structure of the human body for eliminating surgery. Due to the colossal success of medical image processing algorithms (machine learning algorithms) in recent years dramatically led to increase in the usage of electronic medical records and diagnostic imaging. In this project, we developed a new machine learning algorithms as applied to medical image analysis, that primarily focusing on convolutional neural networks, and emphasizing clinical aspects of the field. The asset of machine learning in an era of medical big data is that significant hierarchal relationships within the data can be discovered algorithmically without laborious hand-crafting of features. The model basically covers the areas and applications of medical image classification, localization, detection, segmentation, and registration. We conclude by conferring about research obstacles, emerging trends, and possible future directions.

INDEX TERMS Convolution neural networks, medical image analysis, machine learning, deep learning.

List of figures

List of Tables

LIST OF SCREENSHOTS

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Machine learning algorithms have the potential to be invested deeply in all fields of medicine, from drug discovery to clinical decision making, significantly altering the way medicine is practiced. The success of machine learning algorithms at computer vision tasks in recent years comes at an opportune time when medical records are increasingly digitalized. The use of electronic health records (EHR) quadrupled from 11.8% to 39.6% amongst office-based physicians in the US from 2007 to 2012 [1]. Medical images are an integral part of a patient's EHR and are currently analyzed by human radiologists, who are limited by speed, fatigue, and experience. It takes years and great financial cost to train a qualified radiologist, and some health-care systems out source radiology reporting to lower cost countries such as India via tele-radiology. A delayed or erroneous diagnosis causes harm to the patient. Therefore, it is ideal for medical image analysis to be carried out by an automated, accurate and efficient machine learning algorithm. Medical image analysis is an active field of research for machine learning, partly because the data is relatively structured and labelled, and it is likely that this will be the area where patients first interact with functioning, practical artificial intelligence systems. This is significant for two reasons. Firstly, in terms of actual patient metrics, medical image analysis is a litmus test as to whether artificial intelligence systems will actually improve patient outcomes and survival. Secondly, it provides a testbed for human-AI interaction, of how receptive patients will be towards health- altering choices being made, or assisted by a non-human actor.

1.2 Purpose of the project

The tremendous success of machine learning algorithms at image recognition tasks in recent years intersects with a time of dramatically increased use of electronic medical records and

diagnostic imaging. This review introduces the machine learning algorithms as applied to medical image analysis, focusing on convolutional neural networks, and emphasizing clinical aspects of the field.

1.3 Problem statement

In image processing processing techniques used different types of filters and Fourier and discrete transform it increases the complexity the cost of those equipment also high. To know the result of the tumor concerned person has to be there. This diagnosis perform some particular equipment only .

1.4 Solution for the problem statement:

Detection, sometimes known as Computer-Aided Detection is a keen area of study as missing a lesion on a scan can have drastic consequences for both the patient and the clinician. The task for the Kaggle Data Science Bowl of 2017 involved the detection of cancerous lung nodules on CT lung scans. Approximately 2000 CT scans were released for the competition and the winner Fangzhou achieved a logarithmic loss score of 0.399. Their solution used a 3-D CNN inspired by U-Net architecture to isolate local patches first for nodule detection. Then this output was fed into a second stage consisting of 2 fully connected layers for classification of cancer probability. Shin *et al.*evaluated five well-known CNN architectures in detecting thoracoabdominal lymph nodes and Interstitial lung disease on CT scans. Detecting lymph nodes is important as they can be a marker of infection or cancer. They achieved a mediastinal lymph node detection AUC score of 0.95 with a sensitivity of 85% using GoogLeNet, which was state of the art. They also documented the benefits of transfer learning, and the use of deep learning architectures of up to 22 layers, as opposed to fewer layers which was the norm in medical image analysis. Overfeat was a CNN pre-trained on natural images that won the ILSVRC 2013 localization task . Ciompi applied Overfeat to 2-dimensional slices of CT lung scans oriented in the coronal, axial and sagittal planes, to predict the presence of nodules within and around lung

fissures. They combined this approach with simple SVM and RF binary classifiers, as well as a Bag of Frequencies, a novel 3-dimensional descriptor of their own invention.

CHAPTER 2

SYSTEM ARCHITECTURE

2.1 STUDY OF THE SYSTEM

1. Numpy
2. Panda
3. Matplotlib
4. Scikit –learn

1 . Numpy:

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

2.Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with

Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

3.Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

4.Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Sympy**: Symbolic mathematics
- **Pandas**: Data structures and analysis
- Extensions or modules for SciPy care conventionally named SciKits. As such, the module provides learning algorithms and is named scikit-learn.

CHAPTER 3

SYSTEM ANALYSIS

System analysis will be performed to determine if the system id feasible to design information based on policies and plans of the organization and on user requirements and to eliminate the weakness of the present system.

- The new system should be cost effective.
- To augment management, improve services.
- To enhance user/system interface.
- To improve information quality and usability.
- To upgrade system reliability, availability, flexibility and time saving

3.1 EXISTING SYSTEM

- In image processing techniques used different types of filters and Fourier and discrete transform it increases the complexity the cost of those equipment also high.
- To know the result of the tumor concerned person has to be there.
- This diagnosis perform some particular equipment only

.

3.2 PROPOSED SYSTEM

- Input data plays an important role in prediction along with machine learning techniques.
- In the dataset, provided, we have labels from 0 to 4 where the labels of 4 are hardly 13 and when we split the data into train and test, the number become very less which is nothing but noise and can be totally removed from the dataset by using filtering techniques

- It not only helps us in predicting the outcome but also gave us valuable insights about the nature of data, which can be used in future to train our classifiers in a much better way.

.

3.3 LAYERS

- Convolution Layer-Adding Filters for prediction.
- Relu Layer-Removing Missing and negative values.
- Pooling-Keeping depth constant reducing length and breadth.
- Fully Connected-Connecting the working layer with all others layers.
- Report-Final Prediction.

3.4 SYSTEM CONFIGURATION

Hardware Requirements

RAM : 4GB and Higher

Processor : Intel i3 and above

Hard Disk : 500GB: Minimum

Software Requirements

OS : Windows

Python IDE : python 2.7.x and above

Jupyter Notebook

setup tools and pip to be installed for 3.6.x and above

3.5 FEASIBILITY STUDY

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operational Feasibility
- Economical Feasibility

3.5.1 ECONOMIC FEASIBILITY

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

3.5.2 OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.
The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

3.5.3 TECHNICAL FEASIBILITY

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipments have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

CHAPTER 4

SYSTEM REQUIREMENT SPECIFICATION

A **Software Requirements Specification** (**SRS**) – a requirements specification for a software system – is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional_requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development life cycle domain, typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

- **Business requirements** describe in business terms *what* must be delivered or accomplished to provide value.
- **Product requirements** describe properties of a system or product (which could be one of Several ways to accomplish a set of business requirements.)
- **Process requirements** describe activities performed by the developing organization. For instance, process requirements could specify specific methodologies that must be followed, and constraints that the organization must obey.

Product and process requirements are closely linked. Process requirements often specify the activities that will be performed to satisfy a product requirement. For example, a maximum development cost requirement (a process requirement) may be imposed to help achieve a maximum sales price requirement (a product requirement); a requirement that the product be

maintainable (a Product requirement) often is addressed by imposing requirements to follow particular development styles

PURPOSE

A systems engineering, a **requirement** can be a description of *what* a system must do, referred to as a Functional Requirement. This type of requirement specifies something that the delivered system must be able to do. Another type of requirement specifies something about the system itself, and how well it performs its functions. Such requirements are often called Non-functional requirements, or 'performance requirements' or 'quality of service requirements.' Examples of such requirements include usability, availability, reliability, supportability, testability and maintainability.

A collection of requirements define the characteristics or features of the desired system. A 'good' list of requirements as far as possible avoids saying how the system should implement the requirements, leaving such decisions to the system designer. Specifying how the system should be implemented is called "implementation bias" or "solution engineering". However, implementation constraints on the solution may validly be expressed by the future owner, for example for required interfaces to external systems; for interoperability with other systems; and for commonality (e.g. of user interfaces) with other owned products.

In software engineering, the same meanings of requirements apply, except that the focus of interest is the software itself.

4.1 FUNCTIONAL REQUIREMENTS

Accuracy of model should be high then only we can get perfect results

Need to be analyze the data and remove the unwanted data, if there is any missing values there need to remove those missing values or else has to put suitable for it

Feature selection is the major part of the data analysis get the perfect feature to build model.

4.2 NON FUNCTIONAL REQUIREMENTS

The major non-functional Requirements of the system are as follows

Usability

The system is designed with completely automated process hence there is no or less user intervention.

Reliability

The system is more reliable because of the qualities that are inherited from the chosen platform java. The code built by using java is more reliable.

Performance

This system is developing in the high level languages and using the advanced front-end and back-end technologies it will give response to the end user on client system with in very less time.

Supportability

The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is having JVM, built into the system.

Implementation

The system is implemented in web environment using struts framework. The apache tomcat is used as the web server and windows xp professional is used as the platform.

Interface the user interface is based on Struts provides HTML Tag

4.3 Hardware Requirements:

- RAM: 4GB and Higher
- Processor: Intel i3 and above
- Hard Disk: 500GB: Minimum

4.4 Software Requirements:

- OS: Windows or Linux
- Python IDE : python 2.7.x and above
- Jupyter IDE
- Setup tools and pip to be installed for 3.6 and above
- Language : Python Scripting

4.4.1. INTRODUCTION TO Python

✓ Python

What Is A Script?

Up to this point, I have concentrated on the interactive programming capability of Python. This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

Scripts are reusable

Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor. Then you can execute each of the individual versions. In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor

Just about any text editor will suffice for creating Python script files.

You can use Microsoft Notepad, Microsoft WordPad, Microsoft Word, or just about any word processor if you want to.

Difference between a script and a program

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, where as the applications they control are traditionally compiled to native machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived(e.g., compiled)

Python

what is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don't know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to "big name" languages. Hopefully I can explain it for you.

Python concepts

If your not interested in the the hows and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it's a great one to start programming with.

• Open source general-purpose language.

• Object Oriented, Procedural, Functional

• Easy to interface with C/ObjC/Java/Fortran

• Easy-ish to interface with C++ (via SWIG)

• Great interactive environment

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Dynamic vs Static

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of "thing" each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a "float" type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn't require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn't matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it's more accurate to keep track of decimals so it automatically calculates the result as a floating point number

Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.
Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

- Numbers
- String
- List
- Tuple
- Dictionary

Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as **read-only** lists.

Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .py files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

20 Python libraries

1. Requests. The most famous http library written by kenneth reitz. It's a must have for every python developer.

2. Scrapy. If you are involved in webscraping then this is a must have library for you. After using this library you won't use any other.

3. wxPython. A gui toolkit for python. I have primarily used it in place of tkinter. You will really love it.

4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

5. SQLAlchemy. A database library. Many love it and many hate it. The choice is yours.

6. BeautifulSoup. I know it's slow but this xml and html parsing library is very useful for beginners.

7. Twisted. The most important tool for any network application developer. It has a very beautiful api and is used by a lot of famous python developers.

8. NumPy. How can we leave this very important library ? It provides some advance math functionalities to python.

9. SciPy. When we talk about NumPy then we have to talk about scipy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

10. matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.

11. Pygame. Which developer does not like to play games and develop them ? This library will help you achieve your goal of 2d game development.

12. Pyglet. A 3d animation and game creation engine. This is the engine in which the famous python port of minecraft was made

13. pyQT. A GUI toolkit for python. It is my second choice after wxpython for developing GUI's for my python scripts.

14. pyGtk. Another python GUI library. It is the same library in which the famous Bittorrent client is created.

15. Scapy. A packet sniffer and analyzer for python made in python.

16. pywin32. A python library which provides some useful methods and classes for interacting with windows.

17. nltk. Natural Language Toolkit – I realize most people won't be using this one, but it's generic enough. It is a very useful library if you want to manipulate strings. But it's capacity is beyond that. Do check it out.

18. nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

19. SymPy. SymPy can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

20. IPython. I just can't stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

Numpy

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

• Offers Matlab-ish capabilities within Python

• Fast array operations

• 2D arrays, multi-D arrays, linear algebra etc.

matplotlib

• High quality plotting library.

Python class and objects

These are the building blocks of OOP. class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently "the big thing" in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me.

Having messed around with BASIC in the early '90s, I realized functions were just like subroutines so there wasn't much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you'll realize that it's actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it's not mandatory to use objects in your code in a way that works best; maybe you don't need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.

As you've already seen, Python can do just fine with functions. Unlike languages such as Java, you aren't tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here's a brief list of Python OOP ideas:

• The class statement creates a class object and gives it a name. This creates a new namespace.

• Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: ClassName.Attribute.

• Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.

• Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copies part comes in.

• Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.

• Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

Inheritance

First off, classes allow you to modify a program without really making changes to it.

To elaborate, by subclassing a class, you can change the behavior of the program by simply adding new components to it rather than rewriting the existing components.

As we've seen, an instance of a class inherits the attributes of that class.

However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses.

The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all.

Operator Overloads

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python's object model and the object interfaces are more consistent to Python's built-in objects, hence overloading is easier to learn and use.

User-made classes can override nearly all of Python's built-in operation methods

Exceptions

I've talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program's flow, either intentionally or due to errors.

They are special events that can occur due to an error, e.g. trying to open a file that doesn't exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don't occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python.

Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

• Accessing a non−existent dictionary key will raise a KeyError exception.

• Searching a list for a non−existent value will raise a ValueError exception

. • Calling a non−existent method will raise an AttributeError exception.

• Referencing a non−existent variable will raise a NameError exception.

• Mixing datatypes without coercion will raise a TypeError exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files.

This is the most common way to use exceptions. When programming with the Python command line interpreter, you don’t need to worry about catching exceptions.

Your program is usually short enough to not be hurt too much if an exception occurs.

Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.

It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don’t put any overhead on the code processing.

Because exceptions aren’t supposed to happen very often, they aren’t processed until they occur.

Exceptions can be thought of as a special form of the if/elif statements. You can realistically do the same thing with if blocks as you can with exceptions.

However, as already mentioned, exceptions aren’t processed until they occur; if blocks are processed all the time.

Proper use of exceptions can help the performance of your program.

The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing.

Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions

I won't spend too much time talking about this, but Python does allow for a programmer to create his own exceptions.

You probably won't have to do this very often but it's nice to have the option when necessary.

However, before making your own exceptions, make sure there isn't one of the built-in exceptions that will work for you.

They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free.

Making your own exceptions involves object-oriented programming, which will be covered in the next chapter

. To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class.

To make a custom exception, simply inherit the base exception and define what it will do.

Python modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a *module*; definitions from a module can be *imported* into other modules or into the *main* module.

Testing code

As indicated above, code is usually developed in a file using an editor.

To test the code, import it into a Python session and try to run it.

Usually there is an error, so you go back to the file, make a correction, and test again.

This process is repeated until you are satisfied that the code works. T

he entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

To carry out that specific task, the function might or might not need multiple inputs. When the task is carried out, the function can or can not return one or more values.

There are three types of functions in python:

help(),min(),print().

Python Namespace

Generally speaking, a **namespace** (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a namespacing system from daily life, i.e. the naming of people in firstname and familiy name (surname).

An example is a network: each network device (workstation, server, printer, ...) needs a unique name and address. Yet another example is the directory structure of file systems.

The same file name can be used in different directories, the files can be uniquely accessed via the pathnames.
Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace.

This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs.

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

- **global names** of a module
- **local names** in a function or method invocation
- **built-in names**: this namespace contains built-in functions (e.g. abs(), cmp(), ...) and built-in exception names

Garbage Collection

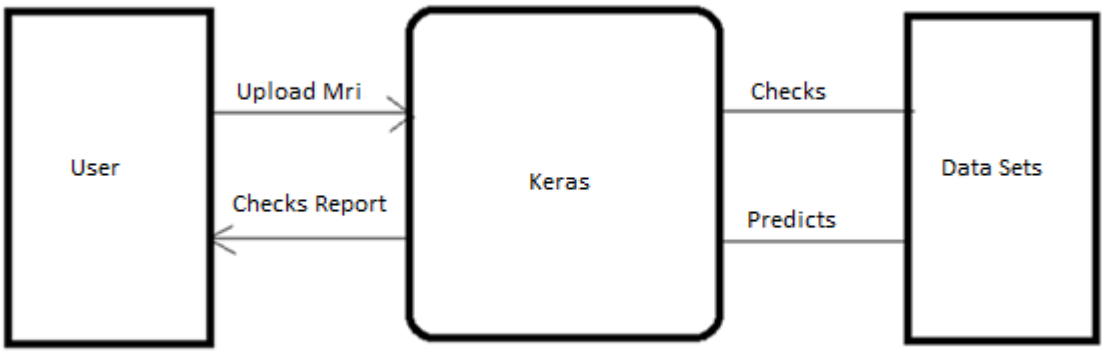
Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

A System Architecture is a conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. System Architecture is abstract, conceptualization-oriented, global, and focused to achieve the mission and life cycle concepts of the system. It also focuses on high level structure in systems and system elements. It addresses the architectural principles, concepts, properties, and characteristics of the system-of-interest. It may also be applied to more than one system, in some cases forming the common structure, pattern, and set of requirements for classes or families of similar or related systems. System architecture conveys the informational content of the elements comprising a system, the relationships among those elements, and the rules governing those relationships. The architectural components and set of relationships between these components that an architecture description may consist of hardware, software, documentation, facilities, manual procedures, or roles played by organizations or people.



5.1.1: Architecture of Brain Tumor Detection

5.2 DATA FLOW DIAGRAM

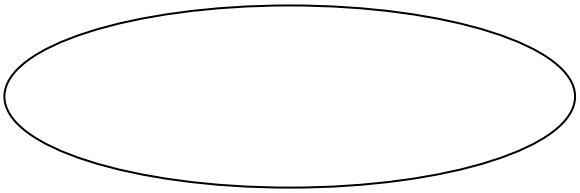
A **data-flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design). On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process. The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

A DFD is also known as a “bubble chart” has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

5.2.1 DFD Symbols

In the DFD, there are four symbols

1. A Square defines a source or destination of system data.
2. An arrow identifies data flow. It is the pipeline through which the information flows.
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store ,data at rest or a temporary repository of data



Process that transforms data flow

Source or destination of data

Data flow

Date store

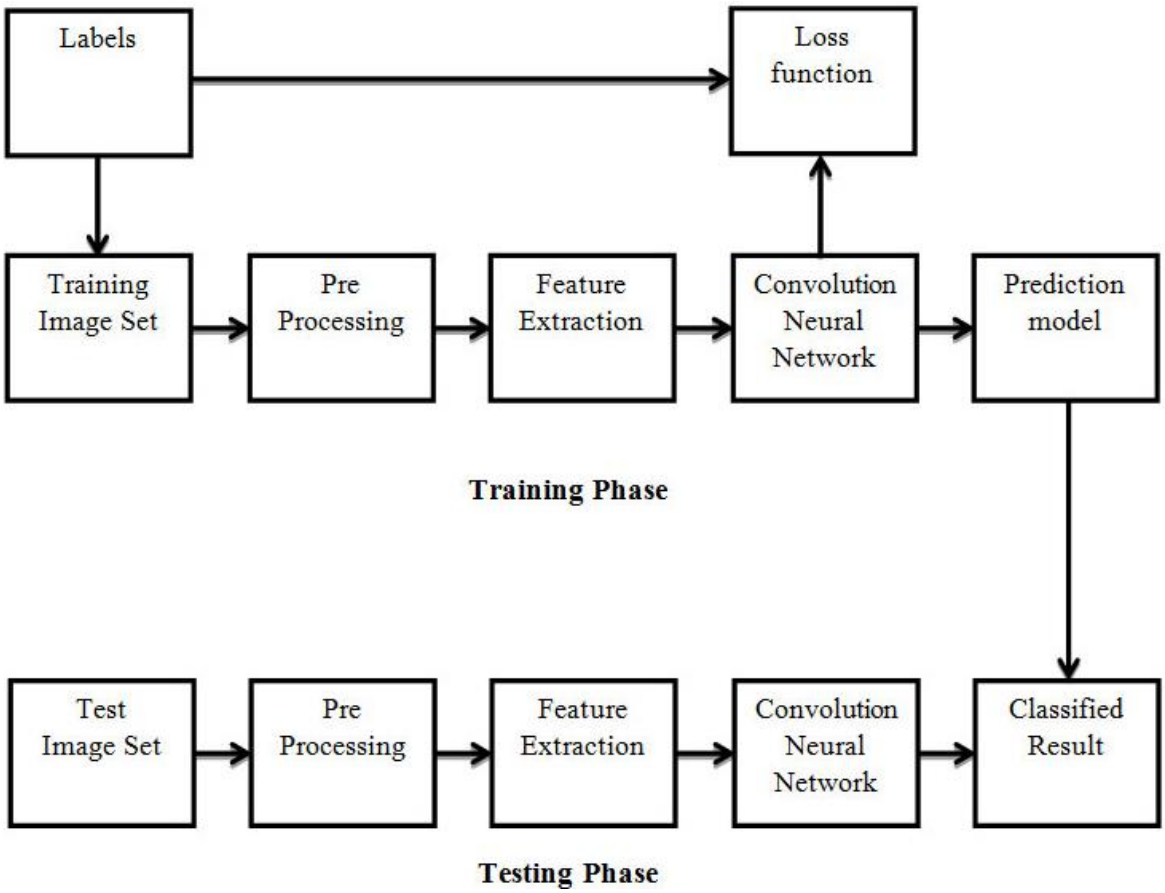


Figure: 5.2.2 DFD for Brain Tumor Detection

5.3.1 UML DIAGRAMS

The Unified Modeling Language (UML) is a standard language for writing software blue prints. The UML is a language for

- Visualizing
- Specifying
- Constructing
- Documenting the artifacts of a software intensive system.

The UML is a language which provides vocabulary and the rules for combining words in that vocabulary for the purpose of communication. A modeling language is a language

whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modeling yields an understanding of a system.

5.3.2 Building Blocks of the UML

The vocabulary of the UML encompasses three kinds of building blocks:

- Things
- Relationships
- Diagrams

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.

5.3.3 Things in the UML

There are four kinds of things in the UML:

- Structural things
- Behavioral things
- Grouping things
- Annotational things

➢ **Structural things** are the nouns of UML models. The structural things used in the project design are:

- First, a **class** is a description of a set of objects that share the same attributes, operations, relationships and semantics.

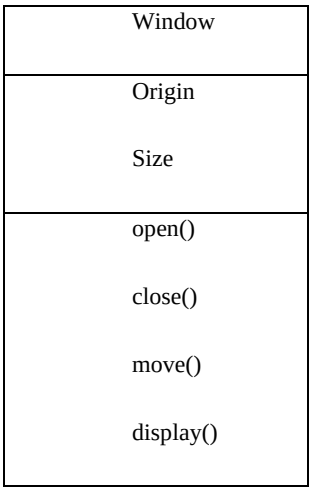


Fig: Classes

- Second, a **use case** is a description of set of sequence of actions that a system performs that yields an observable result of value to particular actor.

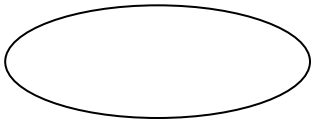


Fig: Use Cases

- Third, a **node** is a physical element that exists at runtime and represents a computational resource, generally having at least some memory and often processing capability.

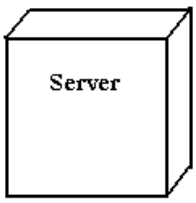


Fig: Nodes

- **Behavioral things** are the dynamic parts of UML models. The behavioral thing used is:

- An **Interaction** is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose. An interaction involves a number of other elements, including messages, action sequences (the behavior invoked by a message, and links (the connection between objects).

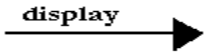


Fig: Messages

5.3.4 Relationships in the UML

There are four kinds of relationships in the UML:

- Dependency
- Association
- Generalization
- Realization

➢ A **dependency** is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing (the dependent thing).

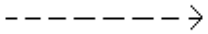


Fig: Dependencies

- An **association** is a structural relationship that describes a set links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts.

Fig: Association

- A **generalization** is a specialization/ generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element(the parent).

Fig: Generalization

- A **realization** is a semantic relationship between classifiers, where in one classifier specifies a contract that another classifier guarantees to carry out.

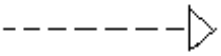


Fig: Realization

5.3.5 ER –Diagram

An entity relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity frame work infrastructure.

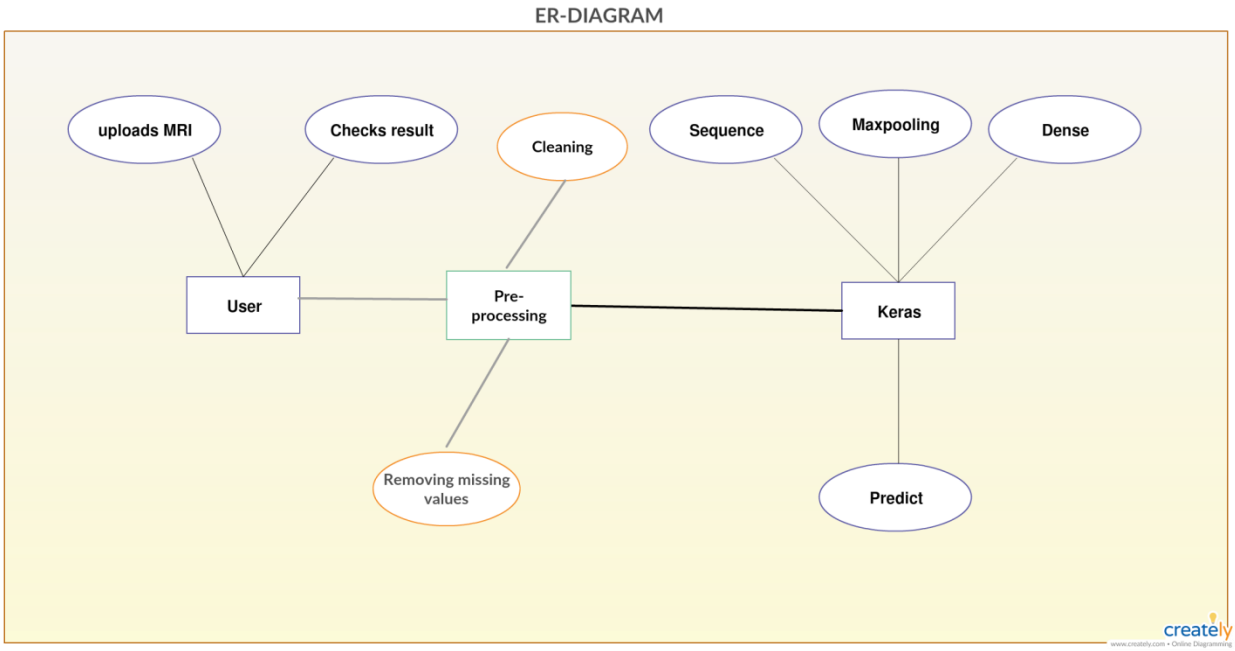


Figure:5.3.5.1 ER DIAGRAM FOR BRAIN TUMOR DETECTION

5.3.6 CLASS DIAGRAM

An object is any person, place, thing, concept, event, screen, or report applicable to your system. Objects both know things (they have attributes) and they do things (they have methods).

A class is a representation of an object and, in many ways, it is simply a template from which objects are created. Classes form the main building blocks of an object-oriented application. Although thousands of students attend the university, you would only model one class, called Student, which would represent the entire collection of students.

Responsibilities

Classes are typically modeled as rectangles with three sections: the top section for the name of the class, the middle section for the attributes of the class, and the bottom section for the methods of the class. Attributes are the information stored about an object, while methods are the things an object or class do. For example, students have student numbers, names, addresses, and phone

numbers. Those are all examples of the attributes of a student. Students also enroll in courses, drop courses, and request transcripts. Those are all examples of the things a student does, which get implemented (coded) as methods. You should think of methods as the object-oriented equivalent of functions and procedures.

CLASS DIAGRAM

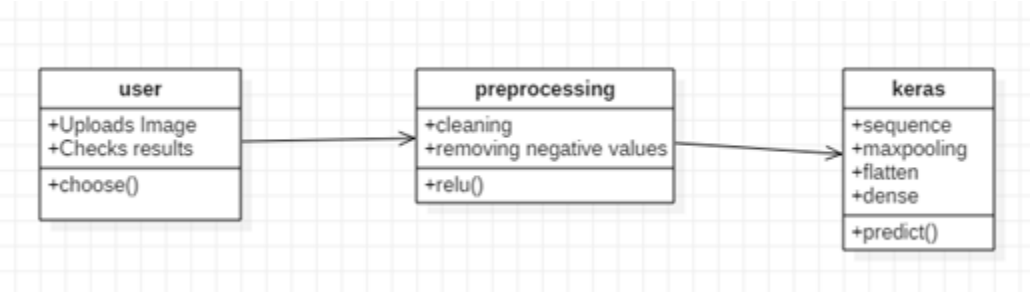


Figure: 5.3.6.1 CLASS DIAGRAM FOR BRAIN TUMOR DETECTION

5.3.7 OBJECT DIAGRAM

An object diagram shows the relationship between a group of objects and their relationships.

Object diagrams represent static snapshot of instance of the things found in class diagram.

Object diagram address the static design or static process view of the system.

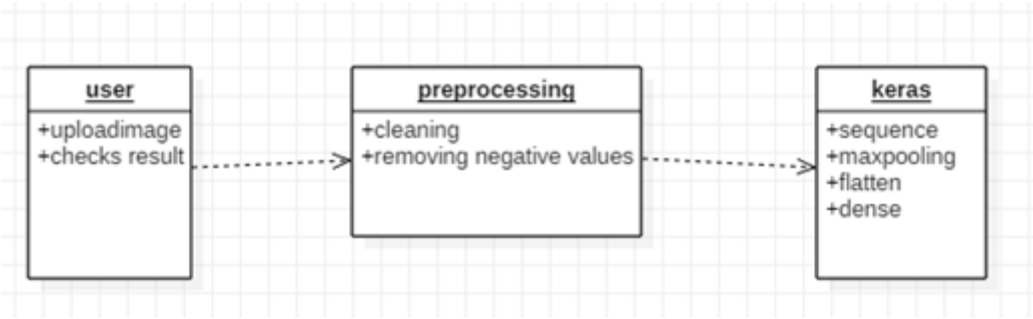


Figure: 5.3.7.1 OBJECT DIAGRAM FOR BRAIN TUMOR DETECTION

5.3.8 USECASE DIAGRAM

A use case diagram is a graph of actors set of use cases enclosed by a system boundary, communication associations between the actors and users and generalization among use cases. The use case model defines the outside (actors) and inside (use case) of the system's behavior.

Use case diagram is quite simple in nature and depicts two types of elements: one representing the business roles and the other representing the business processes.

Fig: an actor in a use case diagram

To identify an actor, search in the problem statement for business terms that portray roles in the system. For example, in the statement "patients visit the doctor in the clinic for medical tests," "doctor" and "patients" are the business roles and can be easily identified as actors in the system.

Use case: A use case in a use case diagram is a visual representation of a distinct business functionality in a system. The key term here is "distinct business functionality." To choose a business process as a likely candidate for modeling as a use case, you need to ensure that the business process is discrete in nature.

As the first step in identifying use cases, you should list the discrete business functions in your problem statement. Each of these business functions can be classified as a potential use case. Remember that identifying use cases is a discovery rather than a creation. As business functionality becomes clearer, the underlying use cases become more easily evident. A use case is shown as an ellipse in a use case diagram (see the below figure).

Fig: use cases in a use case diagram

The above figure shows two uses cases: "Make appointment" and "Perform medical tests" in the use case diagram of a clinic system.

Figure: 5.3.8.1 USECASE DIAGRAM FOR BRAIN TUMOR DETECTION

5.3.9 SEQUENCE DIAGRAM

UML sequence diagrams are used to represent the flow of messages, events and actions between the objects or components of a system. Time is represented in the vertical direction showing the sequence of interactions of the header elements, which are displayed horizontally at the top of the diagram.

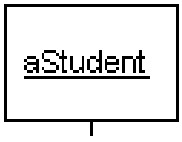
Sequence Diagrams are used primarily to design, document and validate the architecture, interfaces and logic of the system by describing the sequence of actions that need to be performed to complete a task or scenario. UML sequence diagrams are useful design tools because they provide a dynamic view of the system behavior which can be difficult to extract from static diagrams or specifications.

Actor

Represents an external person or entity that interacts with the system

Object

Represents an object in the system or one of its components

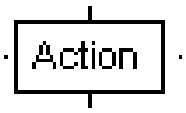


Separator

Represents an interface or boundary between subsystems, components or units (e.g., air interface, Internet, network)

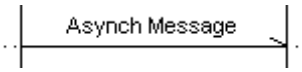
Action

Represents an action taken by an actor, object or unit



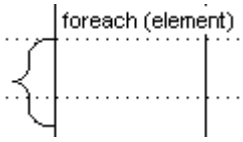
Asynchronous Message

An asynchronous message between header elements



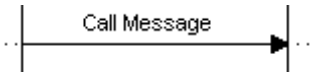
Block

A block representing a loop or conditional for a particular header element



Call Message

A call (procedure) message between header elements



Create Message

A "create" message that creates a header element (represented by lifeline going from dashed to solid pattern)

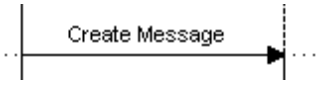
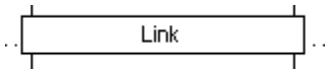
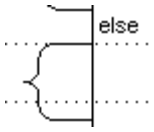


Diagram Link

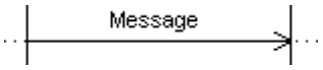
Represents a portion of a diagram being treated as a functional block. Similar to a procedure or function call that abstracts functionality or details not shown at this level. Can optionally be linked to another diagram for elaboration.



Else Block Represents an "else" block portion of a diagram block

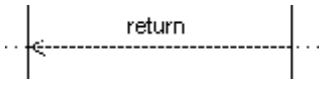


Message:A simple message between header elements



Return Message

A return message between header elements



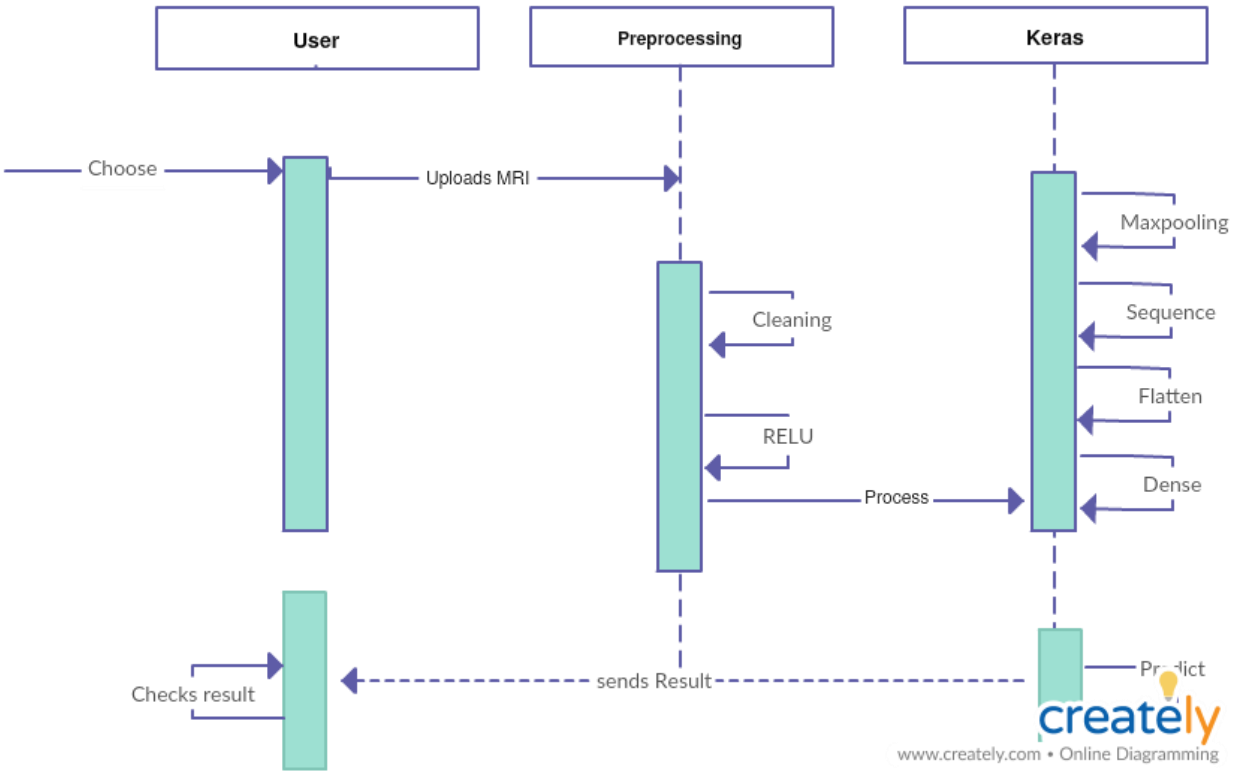


Figure: 5.3.9.1 SEQUENCE DIAGRAM FOR BRAIN TUMOR DETECTION

5.3.10 STATE CHART DIAGRAM

State chart diagram is used to describe the states of different objects in its life cycle. So the emphasis is given on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately. State chart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs.

Before drawing a State chart diagram we must have clarified the following points:

- Identify important objects to be analyzed.
- Identify the states.
- Identify the events.

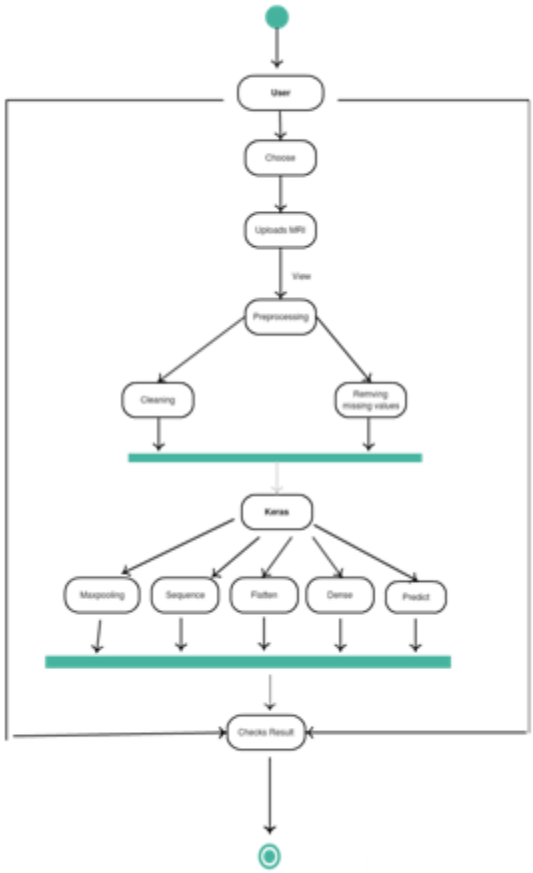


Figure: 5.3.10.1 STATE CHART DIAGRAM FOR BRAIN TUMOR DETECTION

5.3.11 ACTIVITY DIAGRAM

Activity diagrams represent the business and operational workflows of a system. An Activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the particular state.

So, what is the importance of an Activity diagram, as opposed to a State diagram? A State diagram shows the different states an object is in during the lifecycle of its existence in the system, and the transitions in the states of the objects. These transitions depict the activities causing these transitions, shown by arrows.

An Activity diagram talks more about these transitions and activities causing the changes in the object states.

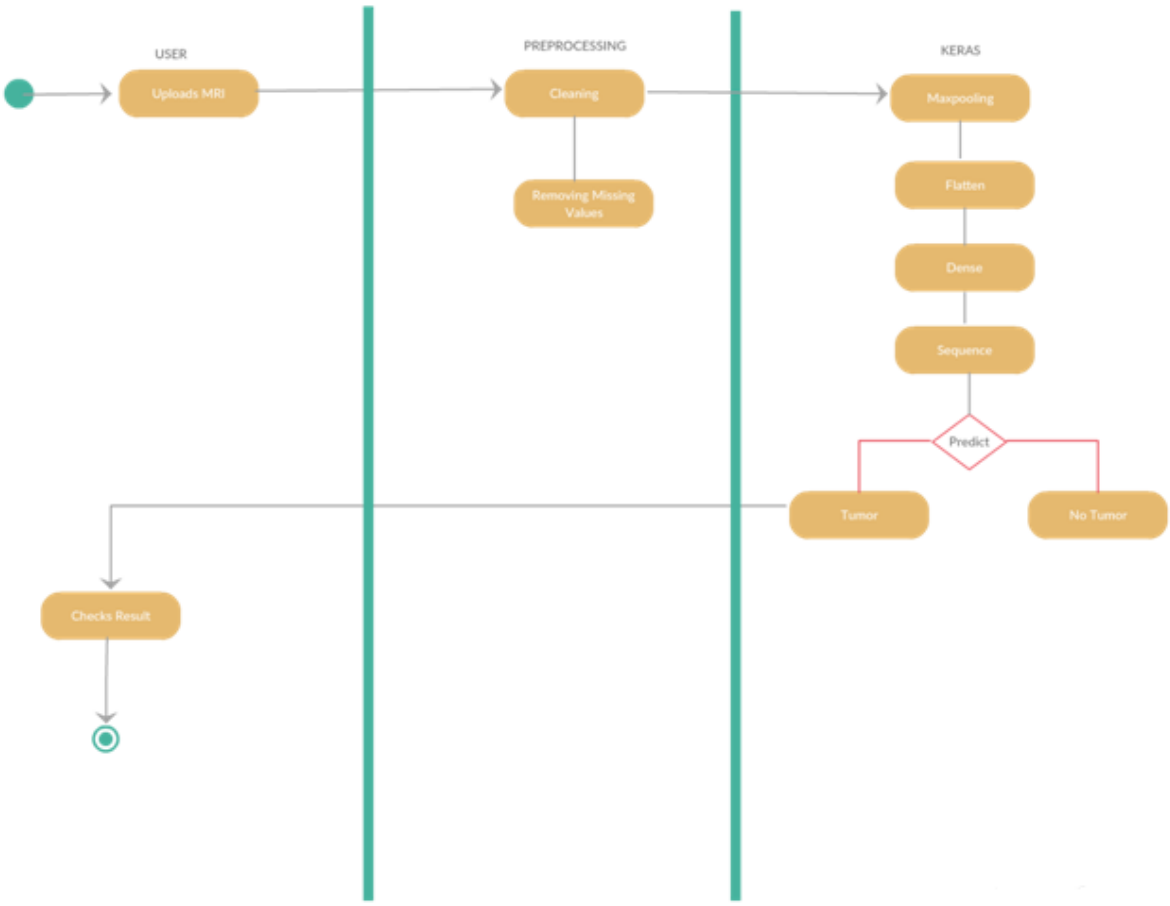


Figure: 5.3.11.1 ACTIVITY DIAGRAM FOR BRAIN TUMOR DETECTION

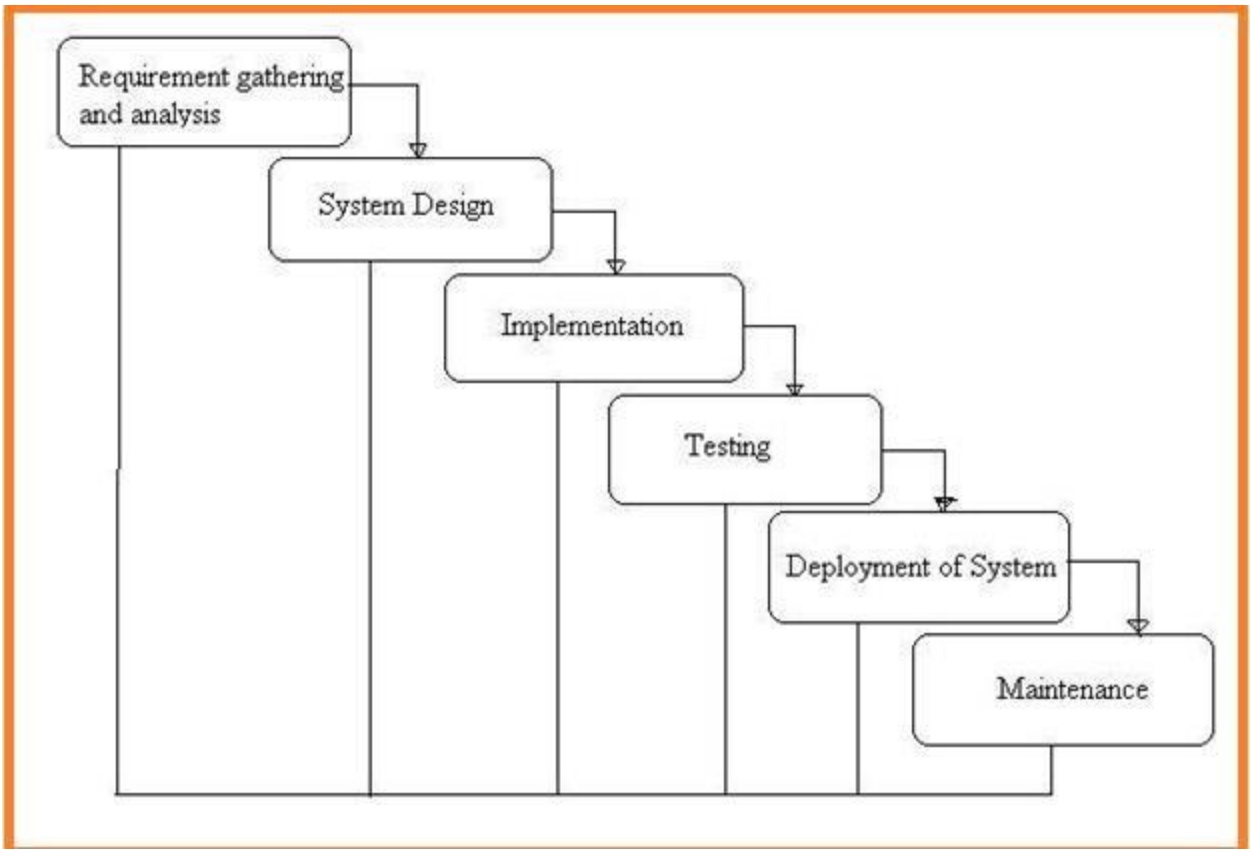


Figure:5.3.12.1 Project SDLC

- Project Requisites Accumulating and Analysis
- Application System Design
- Practical Implementation
- Manual Testing of My Application
- Application Deployment of System
- Maintenance of the Project

Requisites Accumulating and Analysis

It’s the first and foremost stage of any project as an academic leave for requisites amassing we followed of IEEE Journals and Amassed so many IEEE Relegated papers and finally culled a Paper designated by setting and substance importance input and for analysis stage we took referees from the paper and did literature survey of some papers and amassed all the requisites of the project in this stage.

System Design

In System Design, it has been divided into three types like GUI Designing, UML Designing with avails in development of project in facile way with different actor and its utilizer case by utilizer case diagram, flow of the project utilizing sequence, Class diagram gives information about different class in the project with methods that have to be utilized in the project if comes to our project our UML will be utilizable in this way. The third and post import for the project in system design is Data base design where we endeavor to design data base predicated on the number of modules in our project.

Implementation

The Implementation is Phase where we endeavor to give the practical output of the work done in designing stage and most of coding in business logic comes into action in this stage its main and crucial part of the project.

CHAPTER 6

SYSTEM CODING & IMPLEMENTATION

```
# Importing the Keras libraries and packages


from keras.models import Sequential


from keras.layers import Conv2D


from keras.layers import MaxPooling2D


from keras.layers import Flatten


from keras.layers import Dense
classifier = Sequential()#sequencing the input
classifier.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))
classifier.add(Flatten())
classifier.add(Dense(units = 128, activation = 'relu'))
classifier.add(Dense(units = 1, activation = 'sigmoid'))
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
from keras.preprocessing.image import ImageDataGenerator
```

```
train_datagen = ImageDataGenerator(rescale = 1./255,
shear_range = 0.2,
zoom_range = 0.2,
horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
train = train_datagen.flow_from_directory('train',
target_size = (64, 64),
batch_size = 32,
class_mode = 'binary')
test = test_datagen.flow_from_directory('test',
target_size = (64, 64),
batch_size = 32,
class_mode = 'binary')
classifier.fit_generator(train,
steps_per_epoch = 100,
epochs = 25,
validation_data = test,
validation_steps = 40)
import numpy as np
from keras.preprocessing import image
test_image = image.load_img('C:/Users/LENOVO/Desktop/project/train/abnormal/55.jpg', target_size =
(64, 64))
```

```
test_image


test_image = image.img_to_array(test_image)

test_image = np.expand_dims(test_image, axis = 0)

result = classifier.predict(test_image)

#doc=docx.Document('report.docx')

train.class_indices

if result[0][0] == 1:

    print("Tumor")

else:

    print("no Tumor")
```

CHAPTER 7

OUTPUT SCREENS

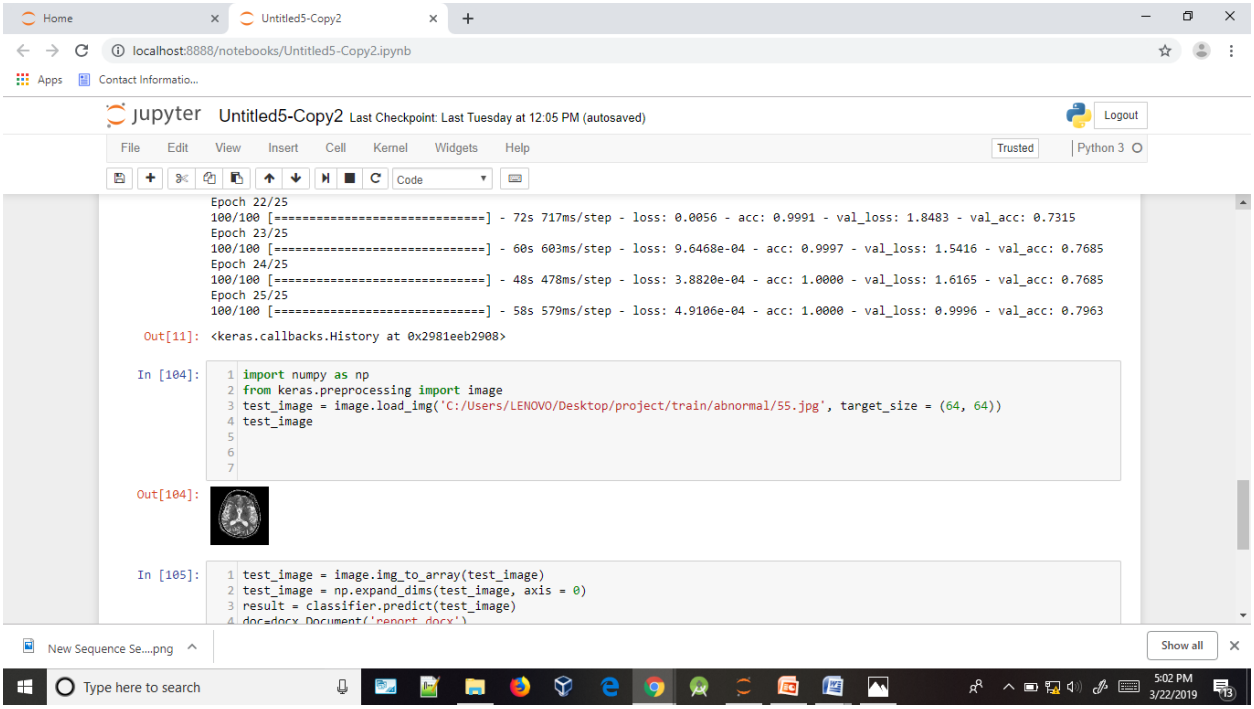


Fig 7.1 Input Screen

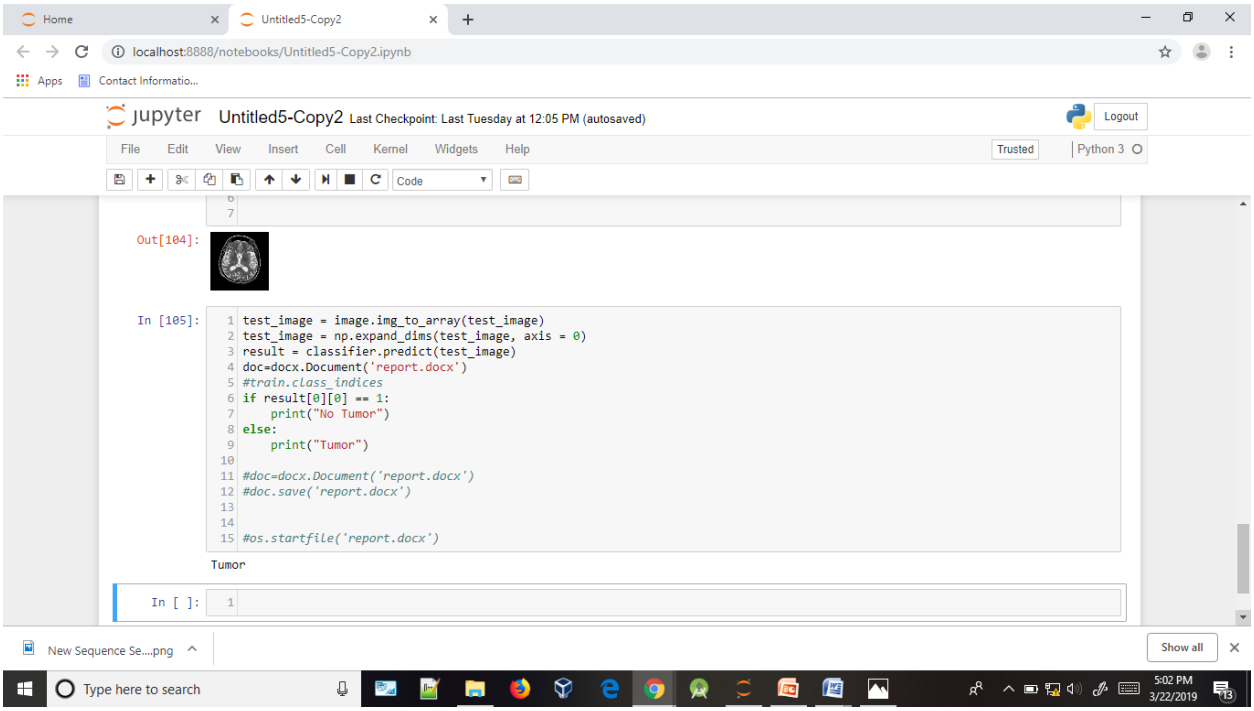


Fig 7.2 Output Screen

CHAPTER 8

TESTING

Testing is the process of evaluating a system or its components with the intent to find that whether it is satisfied the requirements or not. Testing is executing a system in order to identify any gaps or bugs, missing requirements in contrary to the actual desire or requirements.

The following are the testing objectives:

- ➢ Testing is a process executing a program with the intent of finding an error.
- ➢ A good test has a high probability of finding an as yet undiscovered error.
- ➢ A successful test is one that uncovers an as yet undiscovered error.

8.1 TESTING METHODOLOGIES

8.1.1 Integration Testing

Modules integrated by moving down the program design hierarchy. Can use depth first or breadth first top down integration verifies major control and decision points early in design process. Top-level structure tested most. Depth first implementation allows a complete function to be implemented, tested and demonstrated and does depth first implementation of critical function early.

Top down integration forced (to some extend) by some development tools in programs with graphical user interfaces. Begin construction and testing with atomic modules (lowest level modules)

Bottom up integration testing as its name implies being construction and testing with atomic modules. Because modules are integrated from the bottom up, processing required for module subordinate to a given level is always available and the need for stubs is eliminated.

8.1.2 System testing

Once the software product is developed, it is thoroughly tested and it is delivered to the users. Now, it has to be tested by deploying it on the system i.e., to what the given software is comfortable to the environment. The software engineer should consider these issues during early stages of software development to release him from the problems which are encountered after completion of the software. Hence, the tests conducted to ensure that the software is comfortable with the system, where it is deployed is referred as “System Testing”.

8.1.3 Unit testing

This type of testing is performed by developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is different from the test data of the quality assurance team. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

8.1.4 Validation testing

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements. It ensures that the product actually meets the client’s needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed on appropriate environment.

8.2 DESIGN OF TEST CASES AND SCENARIOS

Test cases involve a set of steps, conditions, and inputs that can be used while performing testing tasks. The main intent of this activity is to ensure whether software passes or fails in terms of its functionality and other aspects. There are many types of test cases such as functional, negative, error, logical test cases, physical test cases, UI test cases, etc. Furthermore, test cases are written to keep track of the testing coverage of software. Generally, there are no formal templates that can be used during test case writing.

8.2.1 Test case

A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test cases act as the starting point for the test execution and after applying a set of input values the application has definite outcome and leaves the system at some end point or also known as execution post condition.

8.2.2 Test scenario

It is a one line statement that notifies what area in the application will be tested. Test scenarios are used to ensure that all process flows are tested from end to end. A particular area of an application can have as little as one test scenario to a few hundred scenarios depending on the magnitude and complexity of the application.

The terms 'test scenario' and 'test cases' are used interchangeably, however a test scenario has several steps, whereas a test case has a single step. Viewed from this perspective, test scenarios are test cases, but they include several test cases and the sequence that they should be executed. Apart from this, each test is dependent on the output from the previous test.

CHAPTER 9

CONCLUSION

9.1 CONCLUSION

A recurring theme in machine learning is the limit imposed by the lack of labelled datasets, which hampers training and task performance. Conversely, it is acknowledged that more data improves performance, shows using an internal Google dataset of 300 million images. In general computer vision tasks, attempts have been made to circumvent limited data by using smaller filters on deeper layers, with novel CNN architecture combinations, or hyper parameter optimization.

In medical image analysis, the lack of data is two-fold and more acute: there is general lack of publicly available data, and high quality labelled data is even scarcer. Most of the datasets presented in this review involve fewer than 100 patients. Yet the situation may not be as dire as it seems, as despite the small training datasets, the papers in this review report relatively satisfactory performance in the various tasks. The question of how many images are necessary for training in medical image analysis was partially answered by Cho. He ascertained the accuracy of a CNN with Google Net architecture in classifying individual axial CT images into one of 6 body regions: brain, neck, shoulder, chest, abdomen and pelvis. With 200 training images, accuracies of 88-98% were achieved on a test set of 6000 images. While categorization into various body regions is not a realistic medical image analysis task, his report does suggest that the problem may be surmountable. Being able to accomplish classification with a small dataset is possibly due to the general intrinsic image homogeneity across different patients, as opposed to the near-infinite variety of natural images, such as a dog in various breeds, colors and poses.

VAEs and GANS, being generative models, may sidestep the data paucity problem, by creating synthetic medical data. This was done by Guibas and Virdi, who used a 2 stage GAN to segment and then generate retinal fundus images successfully. Their work was built on the research of Costa, which first described using GANs to generate retinal fundus images. Aside from synthetic

data generation, GANs have been used in brain MRI segmentation as well by Moeskops, Kamnitsas and Alex.

Data or class imbalance in the training set is also a significant issue in medical image analysis. This refers to the number of images in the training data being skewed towards normal and non-pathological images. Rare diseases are an extreme example of this and can be missed without adequate training examples. This data imbalance effect can be ameliorated by using data augmentation to generate more training images of rare or abnormal data, though there is risk of over fitting. Aside from data-level strategies, algorithmic modification strategies and cost sensitive learning have also been studied.

An important, non-technical challenge is the public reception towards their health results being studied by a non- human actor. This situation is not helped by the apocalyptic artificial intelligence scenarios painted by some. Machine learning algorithms have surpassed human performance in image recognition tasks, and it is likely that they will perform better than humans in medical image analysis as well. Indeed, some of the papers in this review report that dermatologists and radiologists have already been bested by machine learning. Yet the question regarding legal and moral culpability arises when a patient is misdiagnosed, or suffers morbidity as a result of AI or AI-assisted medical management. This is accentuated by our inability to fully explain how the black- box of machine algorithms works. However, it is likely that our relationship will continue evolve and recalibrate as AI-based technologies mature and inexorably permeate different facets of our lives.

9.2 Further Enhancements

The traditional applications for medical image analysis were discussed in Section 3. New areas of research include prognostication, content-based image retrieval image report or caption generation and manipulation of physical objects with LSTMs and reinforcement learning involving surgical robots. A few innovative applications that span across traditional medical image analysis categories are described below.

An interesting application was reported by Nie in which GANs were used to generate CT brain images from MRI images. This is remarkable, as it means that patients can potentially avoid the ionizing radiation from a CT scanner altogether, lowering cost and improving patient safety. Nie also exploited the ability of GANs to generate improved, higher resolution images from native images and reduced the blurriness in the CT images. A useful extension of resolution improvement techniques would be applying them to generate MRI images of higher quality. High quality MRI images require high tesla (and correspondingly costlier) MRI scanners. Algorithmically- generated high quality MRI images on a lower field-strength scanner would thus lower healthcare costs.

Chang demonstrated a novel application in the nascent area of radio-genomics, which uses radiological images to predict the underlying molecular origin of a tissue. He first used an auto encoder to learn latent features from MRI images of Glioblastoma Multiform (GBM), a malignant brain tumor, from The Cancer Genome Atlas Glioblastoma Multiforme (TCGA-GBM) data collection. The learned features were then fed into a fully connected classifier layer to classify a MRI scan into one of 4 known molecular sub-types of GBM. Although still early, Chang's work could potentially diagnose a GBM sub-type and obviate the need for surgical biopsy and molecular assays. The generalizability of this technique to tumors elsewhere in the body is also promising. Coudray accomplished an analogous task, but used histopathological images to classify lung cancer sub- types, and to predict common genetic mutations. Knowing the genetic mutations is helpful in prognosticating length of survival and guiding the choice of chemotherapy. Their method outperforms a human pathologist, and the prediction of genetic mutations had AUC scores of between 0.73 to 0.86. Tsochatzidis described an original work combining content-based image retrieval (CBIR) and computer aided diagnosis (CADx). In essence, their model segmented a lesion on a query image, and compared this to the segmented lesions in their database, consisting of 400 Regions of interest derived from the Digital Database for Screening Mammography (DDSM). The basis of comparison was the Euclidean distances between the representation vectors of the query lesion and database lesions. The model then outputs both reference images and a likelihood of a lesion being benign or malignant. They reported that their combined CBIR and CADx method resulted in state of the art prediction accuracy of 81%. These examples highlight how the field of machine learning in medical image

analysis is changing rapidly, and that there may still be numerous applications which have not been conceived of yet.

.

CHAPTER 10

BIBLIOGRAPHY

For software installation:

https://www.anaconda.com/download/

https://www.python.org/downloads/release/python-360/

Modules:

- ✓ Install numpy
- ✓ Install pandas
- ✓ Install Matplotlib
- ✓ Install scikit – learn

References:

C. Wang, X. Dong, F. Zhou, L. Cao, and C.-H. Chi, "Coupled attribute similarity learning on categorical data," IEEE TNNLS, vol. 26, no. 4, pp. 781–797, 2015.

[2] S. Jian, L. Cao, K. Lu, and H. Gao, "Unsupervised coupled metric similarity for non-iid categorical data," IEEE TKDE, 2018.

[3] C. Zhu, L. Cao, Q. Liu, J. Yin, and V. Kumar, "Heterogeneous metric learning of categorical data with hierarchical couplings," IEEE TKDE, 2018.

[4] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," IEEE TPAMI, vol. 35, no. 8, pp. 1798–1828, 2013.

[5] L. Cao, Y. Ou, and S. Y. Philip, "Coupled behavior analysis with applications," IEEE TKDE, vol. 24, no. 8, pp. 1378–1392, 2012.

[6] L. Cao, “Coupling learning of complex interactions,” Information Processing & Management, vol. 51, no. 2, pp. 167–186, 2015.

[7] A. Foss and O. R. Za¨ıane, “A parameterless method for efficiently discovering clusters of arbitrary shape in large datasets,” in Pro- ceedings of ICDM. IEEE, 2002, pp. 179–186.

[8] A. Aizawa, “An information-theoretic perspective of tf–idf mea- sures,” Information Processing & Management, vol. 39, no. 1, pp. 45–65, 2003.

[9] A. Ahmad and L. Dey, “A method to compute distance between two categorical values of same attribute in unsupervised learning for categorical data set,” Pattern Recognition Letters, vol. 28, no. 1, pp. 110–118, 2007. [9] C. Park, D. Kim, J. Oh, and H. Yu, “Predicting user purchase in ecommerce by comprehensive feature engineering and decision boundary focused under-sampling,” in RecSys, 2015.

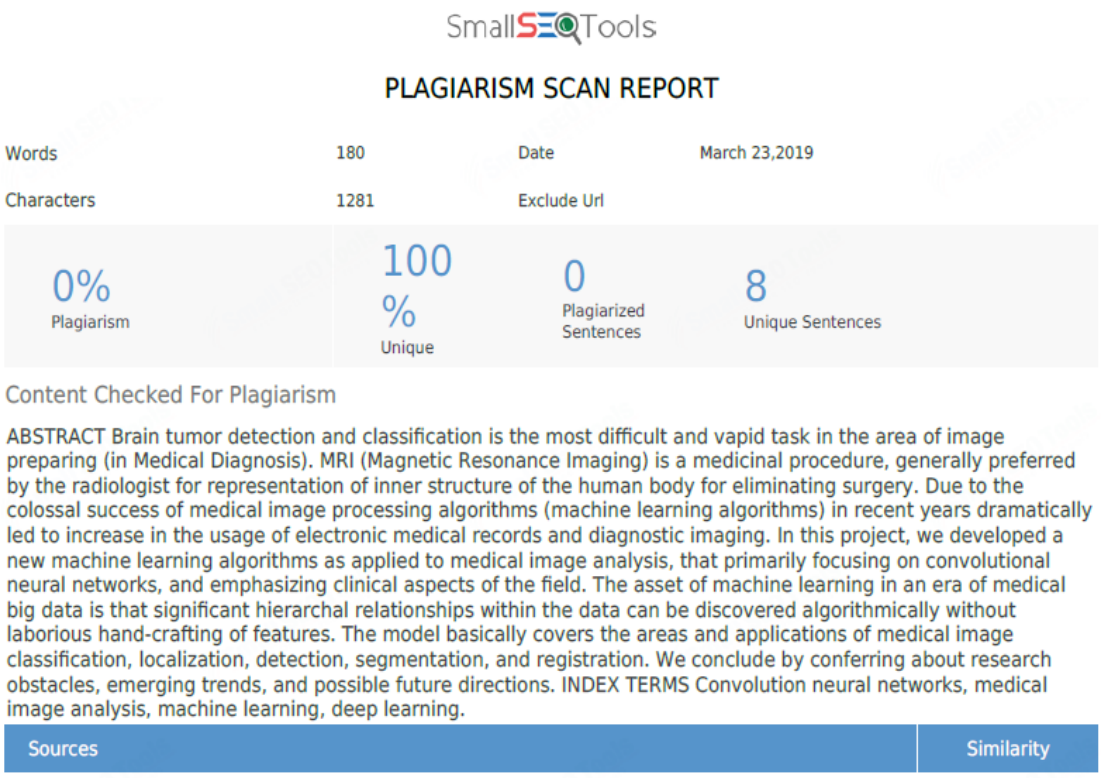
[10] Y. Dong, D. Tao, X. Li, J. Ma, and J. Pu, “Texture classification and retrieval using shearlets and linear regression,” IEEE transactions on cybernetics, vol. 45, no. 3, pp. 358–369, 2015.

[11] X. Li, G. Cui, and Y. Dong, “Refined-graph regularization-based nonnegative matrix factorization,” ACM Transactions on Intelligent Systems and Technology (TIST), vol. 9, no. 1, p. 1, 2017.

[12] J. Lee, M. Podlaseck, E. Schonberg, R. Hoch, and S. Gomory, “Analysis and visualization of metrics for online merchandising,” in WEBKDD’99 Workshop.

CHAPTER 10

PLAGIARISM



SmallSEOTools

PLAGIARISM SCAN REPORT

Words	180	Date	March 23,2019
Characters	1281	Exclude Url	

0%	100 %	0	8
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

ABSTRACT Brain tumor detection and classification is the most difficult and vapid task in the area of image preparing (in Medical Diagnosis). MRI (Magnetic Resonance Imaging) is a medicinal procedure, generally preferred by the radiologist for representation of inner structure of the human body for eliminating surgery. Due to the colossal success of medical image processing algorithms (machine learning algorithms) in recent years dramatically led to increase in the usage of electronic medical records and diagnostic imaging. In this project, we developed a new machine learning algorithms as applied to medical image analysis, that primarily focusing on convolutional neural networks, and emphasizing clinical aspects of the field. The asset of machine learning in an era of medical big data is that significant hierarchal relationships within the data can be discovered algorithmically without laborious hand-crafting of features. The model basically covers the areas and applications of medical image classification, localization, detection, segmentation, and registration. We conclude by conferring about research obstacles, emerging trends, and possible future directions. INDEX TERMS Convolution neural networks, medical image analysis, machine learning, deep learning.

Sources	Similarity

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	70	Date	March 23,2019
Characters	470	Exclude Url	

0%	100 %	0	3
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

A recurring theme in machine learning is the limit imposed by the lack of labelled datasets, which hampers training and task performance. Conversely, it is acknowledged that more data improves performance, shows using an internal Google dataset of 300 million images. In general computer vision tasks, attempts have been made to circumvent limited data by using smaller filters on deeper layers, with novel CNN architecture combinations, or hyper parameter optimization.

Sources	Similarity

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	71	Date	September 11,2018
Characters	415	Exclude Url	

0%	100 %	0	3
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

Further Enhancements The project made here is just to ensure that this project could be valid in today real challenging world .Here all the facilities are made. As there is always a room for improvement in any software package, The important thing is that the app should be flexible enough for further modifications. In future, we can authenticate user by their identity number instead of email id's.

Sources	Similarity