1. INTRODUCTION

Accurate segmentation of tumours through medical images is of particular importance as it provides information essential for analysis and diagnosis of cancer as well as for mapping out treatment options and monitoring the progression of the disease. Brain tumours are one of the fatal cancers worldwide and are categorised, depending upon their origin, into primary and secondary tumour types. The most common histological form of primary brain cancer is the glioma, which originates from the brain glial cells and attributes towards 80% of all malignant brain tumours. Gliomas can be of the slow-progressing low-grade (LGG) subtype with a better patient prognosis or are the more aggressive and infiltrative high-grade glioma (HGG) or glioblastoma, which require immediate treatment. These tumours are associated with substantial morbidity, where the median survival for a patient with glioblastoma is only about 14 months with a 5-year survival rate near zero despite maximal surgical and medical therapy.

A timely diagnosis, therefore, becomes imperative for effective treatment of the patients. Magnetic Resonance Imaging (MRI) is a preferred technique widely employed by radiologists for the evaluation and assessment of brain tumours . It provides several complimentary 3D MRI modalities acquired based on the degree of excitation and repetition times, i.e. T1-weighted, post-contrast T1-weighted (T1ce), T2-weighted and Fluid-Attenuated Inversion Recovery (FLAIR).

The highlighted subregions of the tumour across different intensities of these sequences, such as the whole tumour (the entire tumour inclusive of infiltrative oedema), is more prominent in FLAIR and T2 modalities. In contrast, T1 and T1ce images show the tumour core exclusive of peritumoural oedema. It allows for the combinative use of these scans and the complementary information they deliver towards the detection of different tumour subregions.

1.1.Objective of the project:

Automated segmentation of brain tumour from multimodal MR images is pivotal for the analysis and monitoring of disease progression. As gliomas are malignant and heterogeneous, efficient and accurate segmentation techniques are used for the successful delineation of tumours into intra-tumoural classes. Deep learning algorithms outperform on tasks of semantic segmentation as opposed to the more conventional, context-based computer vision approaches. Extensively used for biomedical image segmentation, Convolutional Neural Networks have significantly improved the state-of-the-art accuracy on the task of brain tumour segmentation.

In this paper, we propose an ensemble of two segmentation networks: a 3D CNN and a U-Net, in a significant yet straightforward combinative technique that results in better and accurate predictions. Both models were trained separately on the BraTS-19 challenge dataset and evaluated to yield segmentation maps which considerably differed from each other in terms of segmented tumour sub-regions and were ensembled variably to achieve the final prediction. The suggested ensemble achieved dice scores of 0.750, 0.906 and 0.846 for enhancing tumour, whole tumour, and tumour core, respectively, on the validation set, performing favourably in comparison to the state-of-the-art architectures currently available.

2. LITERATURE SURVEY

A survey of MRI-based medical image analysis for brain tumour studies

MRI-based medical image analysis for brain tumor studies is gaining attention in recent times due to an increased need for efficient and objective evaluation of large amounts of data. While the pioneering approaches applying automated methods for the analysis of brain tumor images date back almost two decades, the current methods are becoming more mature and coming closer to routine clinical application. This review aims to provide a comprehensive overview by giving a brief introduction to brain tumors and imaging of brain tumors first. Then, we review the state of the art in segmentation, registration and modeling related to tumor-bearing brain images with a focus on gliomas. The objective in the segmentation is outlining the tumor including its sub-compartments and surrounding tissues, while the main challenge in registration and modeling is the handling of morphological changes caused by the tumor. The qualities of different approaches are discussed with a focus on methods that can be applied on standard clinical imaging protocols. Finally, a critical assessment of the current state is performed and future developments and trends are addressed, giving special attention to recent developments in radiological tumor assessment guidelines.

Global incidence of malignant brain and other central nervous system tumours by histology, 2003—2007

Previous reports have shown that overall incidence of malignant brain and other central nervous system (CNS) tumors varied significantly by country. The aim of this study was to estimate histology-specific incidence rates by global region and assess incidence variation by histology and age. Using data from the Central Brain Tumor Registry of the United States (CBTRUS) and the International Agency for Research on Cancer's (IARC) Cancer Incidence in Five Continents X (including over 300 cancer registries), we calculated the age-adjusted incidence rates (AAIR) per 100000 person-years and 95% CIs for brain and other CNS tumors overall and by age groups and histology. Brain and other CNS tumors are a significant source of cancer-related morbidity and mortality worldwide. Regional differences in incidence may provide clues toward genetic or environmental causes as well as a foundation for broadening knowledge of their epidemiology.

Gaining a comprehensive understanding of the epidemiology of malignant brain tumors globally is critical to researchers, public health officials, disease interest groups, and clinicians and contributes to collaborative efforts in future research**.**

CBTRUS statistical report: primary brain and central nervous system tumours diagnosed in the United States in 2005--2009

The Central Brain Tumor Registry of the United States (CBTRUS), in collaboration with the Centers for Disease Control (CDC) and National Cancer Institute (NCI), is the largest population-based registry focused exclusively on primary brain and other central nervous system (CNS) tumors in the United States (US) and represents the entire US population. This report contains the most up-to-date population-based data on primary brain tumors (malignant and non-malignant) and supersedes all previous CBTRUS reports in terms of completeness and accuracy. All rates (incidence and mortality) are age-adjusted using the 2000 US standard population and presented per 100,000 population. The average annual age-adjusted incidence rate (AAAIR) of all malignant and non-malignant brain and other CNS tumors was 23.79 (Malignant AAAIR=7.08, non-Malignant AAAIR=16.71).

This rate was higher in females compared to males (26.31 versus 21.09), Blacks compared to Whites (23.88 versus 23.83), and non-Hispanics compared to Hispanics (24.23 versus 21.48). The most commonly occurring malignant brain and other CNS tumor was glioblastoma (14.5% of all tumors), and the most common non-malignant tumor was meningioma (38.3% of all tumors). Glioblastoma was more common in males, and meningioma was more common in females. In children and adolescents (age 0-19 years), the incidence rate of all primary brain and other CNS tumors was 6.14. An estimated 83,830 new cases of malignant and non-malignant brain and other CNS tumors are expected to be diagnosed in the US in 2020 (24,970 malignant and 58,860 non-malignant). There were 81,246 deaths attributed to malignant brain and other CNS tumors between 2013 and 2017. This represents an average annual mortality rate of 4.42. The 5-year relative survival rate following diagnosis of a malignant brain and other CNS tumor was 36.0% and for a non-malignant brain and other CNS tumor was 91.7%.

The 2016 World Health rganization classification of tumours of the central nervous system

The 2016 World Health Organization Classification of Tumors of the Central Nervous System is both a conceptual and practical advance over its 2007 predecessor. For the first time, the WHO classification of CNS tumors uses molecular parameters in addition to histology to define many tumor entities, thus formulating a concept for how CNS tumor diagnoses should be structured in the molecular era.

As such, the 2016 CNS WHO presents major restructuring of the diffuse gliomas, medulloblastomas and other embryonal tumors, and incorporates new entities that are defined by both histology and molecular features, including glioblastoma, IDH-wildtype and glioblastoma, IDH-mutant; diffuse midline glioma, H3 K27M-mutant; RELA fusion-positive ependymoma; medulloblastoma, WNT-activated and medulloblastoma, SHH-activated; and embryonal tumour with multilayered rosettes, C19MC-altered.

The 2016 edition has added newly recognized neoplasms, and has deleted some entities, variants and patterns that no longer have diagnostic and/or biological relevance. Other notable changes include the addition of brain invasion as a criterion for atypical meningioma and the introduction of a soft tissue-type grading system for the now combined entity of solitary fibrous tumor / hemangiopericytoma-a departure from the manner by which other CNS tumors are graded. Overall, it is hoped that the 2016 CNS WHO will facilitate clinical, experimental and epidemiological studies that will lead to improvements in the lives of patients with brain tumors.

Radiotherapy plus concomitant and adjuvant temozolomide for glioblastoma

Glioblastoma, the most common primary brain tumor in adults, is usually rapidly fatal. The current standard of care for newly diagnosed glioblastoma is surgical resection to the extent feasible, followed by adjuvant radiotherapy. In this trial we compared radiotherapy alone with radiotherapy plus temozolomide, given concomitantly with and after radiotherapy, in terms of efficacy and safety.Patients with newly diagnosed, histologically confirmed glioblastoma were randomly assigned to receive radiotherapy alone (fractionated focal irradiation in daily fractions of 2 Gy given 5 days per week for 6 weeks, for a total of 60 Gy) or radiotherapy plus continuous daily temozolomide (75 mg per square meter of body-surface area per day, 7 days

per week from the first to the last day of radiotherapy), followed by six cycles of adjuvant temozolomide (150 to 200 mg per square meter for 5 days during each 28-day cycle). The primary end point was overall survival.

A total of 573 patients from 85 centers underwent randomization. The median age was 56 years, and 84 percent of patients had undergone debulking surgery. At a median follow-up of 28 months, the median survival was 14.6 months with radiotherapy plus temozolomide and 12.1 months with radiotherapy alone. The unadjusted hazard ratio for death in the radiotherapy-plus-temozolomide group was 0.63 (95 percent confidence interval, 0.52 to 0.75; $P<0.001$ by the log-rank test). The two-year survival rate was 26.5 percent with radiotherapy plus temozolomide and 10.4 percent with radiotherapy alone. Concomitant treatment with radiotherapy plus temozolomide resulted in grade 3 or 4 hematologic toxic effects in 7 percent of patients.The addition of temozolomide to radiotherapy for newly diagnosed glioblastoma resulted in a clinically meaningful and statistically significant survival benefit with minimal additional toxicity.

3. SYSYEM ANALYSIS

3.1 Existing System

To automate brain tumour segmentation process author is combining both 3D CNN and UNET algorithms as deep learning is gaining popularity in efficient semantic segmentation of medical images. To further enhance segmentation process author is using combination or ensemble of two deep learning algorithms called CNN and UNET. Both algorithms trained separately on BRATS brain tumour dataset and then predicted output of both algorithms will be merge or map to generate final segmentation and the output generated is giving high dice score after mapping both algorithms segmentation and then predicting final segmented output. Dice score refers to correctly mapping of segmented parts in the image.The task is to develop an automated brain tumour segmentation method, for successful delineation of tumours into intra-tumoural classes with improved efficiency and accuracy in comparison to existing methods.

Existing systems for brain tumor segmentation using CNNs and U-Net have demonstrated promising performance, offering automated and accurate delineation of tumor regions in medical images. These systems have the potential to significantly assist radiologists and clinicians in diagnosis, treatment planning, and monitoring of patients with brain tumors. Continued advancements in this area, including the exploration of novel network architectures and optimization strategies, hold the promise of further enhancing the efficacy and efficiency of brain tumor segmentation algorithms, ultimately benefiting patient care and outcomes.

Disadvantage:

1.Less Accuracy.

3.2. Proposed System:

To implement this project we are using 4 different images and this images are called as FLAIR, T1, T2 and T1CE and the label segmented image. The multi-institutional dataset, acquired from 19 different contributors, contains multimodal MRI scans of each patient, namely T1, T1 contrast-enhanced (T1ce), T2-weighted (T2), and Fluid Attenuated Inversion Recovery (FLAIR), from which the tumoural sub regions are segmented. The data is processed to overcome discrepancies such that they are skull-stripped.

The proposed system for brain tumor segmentation using convolutional neural networks (CNNs) and the U-Net architecture aims to further improve the accuracy, efficiency, and clinical applicability of existing methods. This system builds upon the strengths of CNNs and U-Net while integrating novel techniques and considerations to address specific challenges in brain tumor segmentation.

Moreover, the proposed system emphasizes interpretability and clinical relevance by incorporating uncertainty estimation techniques. By quantifying the uncertainty associated with segmentation predictions, the system provides clinicians with confidence intervals or probabilistic maps, enabling them to make informed decisions based on the reliability of the segmentation results.

Advantage:

1.More Accuracy.

4. MODULES

4.1 Modules names and Explanation:

1.upload BRATS Dataset

2.Generate CNN & UNET Model

3.Upload Test Image & Segmentation

4.Dice Similarity Graph

1.Upload BRATS Dataset:

Upload BRATS Dataset is the first module of our project,it is used to upload the BRATS dataset.

The BRATS (Brain Tumor Segmentation) dataset is a widely used collection of MRI scans of brain tumors. It consists of MRI images along with corresponding ground truth segmentation masks, which label different regions of the brain tumor. These datasets are typically uploaded to a platform or environment where they can be accessed for training and evaluation.

2.Generate CNN & UNET Model:

Generate CNN & UNET Model is the second module of our project,it is used to models are generated and console to see CNN and UNET layer details. we can see models are using different size images to filter them and to get best features from it to build efficient model and now model is generated.

Convolutional Neural Networks (CNNs) and UNet are popular deep learning architectures for image segmentation tasks like brain tumor segmentation. CNNs use convolutional layers to extract features from input images, while UNet is known for its encoder-decoder architecture, which helps capture both local and global features of the image. Training involves feeding the network with input images and their corresponding ground truth masks, optimizing the model parameters to minimize the difference between predicted and ground truth masks.

3.Upload Test Image & Segmentation:

Upload Test Image & Segmentation is the third module of our project and then upload test samples to get segmented output. selecting and uploading 'Sample1' folder and then click on 'Select Folder' button to get below outputtop 4 images are the input images such as FLAIR, T1, T2 and T1CE and 5th image is the predicted image with segmented part showing in red

colour and this algorithm correctly detecting and marking tumour area and now test with other image.

4.Dice Similarity Graph:

To build CNN and UNET model we took 50 epoch or iterations and at each iteration DICE score between training and testing images get better and better and we get final dice score as 0.8 * 100 = 80%. In above graph x-axis represents epoch and y-axis represents dice score.

The Dice similarity coefficient is a metric commonly used to evaluate the performance of segmentation algorithms. It measures the overlap between the predicted segmentation mask and the ground truth mask. A Dice score close to 1 indicates a high level of agreement between the predicted and ground truth masks. By plotting the Dice similarity coefficients for different test images, you can assess the overall performance of the CNN and UNet models across the dataset

5. FEASIBILITY STUDY

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.

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. 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.

5.3 TECHNICAL FEASIBILITY

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.

6. REQUIREMENTS

6.1 SOFTWARE REQUIREMENTS

Windows: Python 3.6.2 or above, PIP and NumPy 1.13.1 Python:

Python is an interpreted, high-level, general purpose programming language created by Guido Van Rossum and first released in 1991, Python's design philosophy emphasizes code Readability with its notable use of significant Whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage collected.

PIP:

It is the package management system used to install and manage software packages written in Python.

NumPy:

NumPy is a general-purpose array-processing package. It provides a highperformance 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:

- 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

Keras:

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or Plaid ML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

OpenCV:

OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by willow garage then Itseez (which was later acquired by Intel). The library is cross platform and free for use under the open source BSD license. OpenCV supports some models from deep learning frameworks like TensorFlow, Torch, PyTorch (after converting to an ONNX model) and Caffe according to a defined list of supported layers. It promotes Open Vision Capsules. which is a portable format, compatible with all other formats.

6.2 HARDWARE REQUIREMENTS

- Processor: Intel core i5 or above.
- 64-bit, quad-core, 2.5 GHz minimum per core
- Ram: 4 GB or more
- Hard disk: 10 GB of available space or more.
- Display: Dual XGA (1024 x 768) or higher resolution monitors
- Operating system: Windows

7. IMPLEMETATION

Python

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

History of Python

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

Why Python was created?

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

Why the name Python?

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

Features of Python

A simple language which is easier to learn

Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax. If you are a newbie, it's a great choice to start your journey with Python.

Free and open-source

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute softwares written in it, you can even make changes to the Python's source code.

Portability

You can move Python programs from one platform to another, and run it without any changes. It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

Extensible and Embeddable

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code. This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

A high-level, interpreted language

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on. Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

Large standard libraries to solve common tasks

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using import MySQLdb . Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

Object-oriented

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.With OOP, you are able to divide these complex problems into smaller sets by creating objects.

Applications of Python:

1. Simple Elegant Syntax

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

```
a = 2

b = 3

sum = a + b

print(sum)
```

2. Not overly strict

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement. Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

3. Expressiveness of the language

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code.

This is just an example. You will be amazed how much you can do with Python once you learn the basics.

4. Great Community and Support

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

7.1 Sample Code:

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import simpledialog
from tkinter import filedialog
import numpy as np
from tkinter.filedialog import askopenfilename
import pickle
import os
import cv2
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

from keras.models import *
from keras.layers import *
from keras.optimizers import *


gui = tkinter.Tk()
gui.title("Brain Tumour Image Segmentation Using Deep Networks")
gui.geometry("1300x1200")

global filename
global model
global X, Y

def dice_coef(y_true, y_pred):
    y_true_f = keras.flatten(y_true)
    y_pred_f = keras.flatten(y_pred)
    intersection = keras.sum(y_true_f * y_pred_f)
    return (2. * intersection + 1) / (keras.sum(y_true_f) +
keras.sum(y_pred_f) + 1)

def dice_coef_loss(y_true, y_pred):
    return -dice_coef(y_true, y_pred)

def getModel(input_size=(64,64,1)):
    inputs = Input(input_size)

    conv1 = Conv2D(32, (3, 3), activation='relu',
padding='same')(inputs)
    conv1 = Conv2D(32, (3, 3), activation='relu',
padding='same')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
```

```
conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(pool1)
   conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

    conv3 = Conv2D(128, (3, 3), activation='relu',
padding='same')(pool2)
    conv3 = Conv2D(128, (3, 3), activation='relu',
padding='same')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)

    conv4 = Conv2D(256, (3, 3), activation='relu',
padding='same')(pool3)
    conv4 = Conv2D(256, (3, 3), activation='relu',
padding='same')(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)

    conv5 = Conv2D(512, (3, 3), activation='relu',
padding='same')(pool4)
    conv5 = Conv2D(512, (3, 3), activation='relu',
padding='same')(conv5)

    up6 = concatenate([Conv2DTranspose(256, (2, 2), strides=(2, 2),
padding='same')(conv5), conv4], axis=3)
    conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(up6)
    conv6 = Conv2D(256, (3, 3), activation='relu',
padding='same')(conv6)

    up7 = concatenate([Conv2DTranspose(128, (2, 2), strides=(2, 2),
padding='same')(conv6), conv3], axis=3)
    conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(up7)
    conv7 = Conv2D(128, (3, 3), activation='relu',
padding='same')(conv7)

    up8 = concatenate([Conv2DTranspose(64, (2, 2), strides=(2, 2),
padding='same')(conv7), conv2], axis=3)
    conv8 = Conv2D(64, (3, 3), activation='relu', padding='same')(up8)
    conv8 = Conv2D(64, (3, 3), activation='relu',
padding='same')(conv8)

    up9 = concatenate([Conv2DTranspose(32, (2, 2), strides=(2, 2),
padding='same')(conv8), conv1], axis=3)
    conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(up9)
    conv9 = Conv2D(32, (3, 3), activation='relu',
padding='same')(conv9)

    conv10 = Conv2D(1, (1, 1), activation='sigmoid')(conv9)

    return Model(inputs=[inputs], outputs=[conv10])
```

```
def uploadDataset():
    global X, Y
    global filename
    text.delete('1.0', END)
    filename = filedialog.askdirectory(initialdir=".")
    text.insert(END,filename+" loaded\n");
    '''
    X = []
    Y = []
    for root, dirs, directory in os.walk(filename):
        for i in range(len(directory)):
            img = cv2.imread(train_directory+"/"+directory[i],0)
            img = cv2.resize(img,(64,64), interpolation =
cv2.INTER_CUBIC)
            X.append(img)
            img = cv2.imread("dataset/label/"+directory[i],0)
            img = cv2.resize(img,(64,64), interpolation =
cv2.INTER_CUBIC)
            Y.append(img)

    X = np.asarray(X)
    Y = np.asarray(Y)
    '''
def generateModel():
    global model
    '''
    global X, Y
    dim = 64
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size
= 0.10, random_state = 1)
    X_train = X_train.reshape(len(X_train),dim,dim,1)
    y_train = y_train.reshape(len(y_train),dim,dim,1)
    X_test = X_test.reshape(len(X_test),dim,dim,1)
    y_test = y_test.reshape(len(y_test),dim,dim,1)
    images = np.concatenate((X_train,X_test),axis=0)
    mask  = np.concatenate((y_train,y_test),axis=0)
    tr = X_train[12]
    yr = y_train[12]
    cv2.imshow('tr',tr)
    cv2.imshow('yr',yr)
    cv2.waitKey(0)
    '''
    model = getModel(input_size=(64,64,1))
    with open('model/model.json', "r") as json_file:
        loaded_model_json = json_file.read()
        model = model_from_json(loaded_model_json)
    json_file.close()
    model.load_weights("model/model_weights.h5")
    model._make_predict_function()
    print(model.summary())

text.insert(END,"CNN & UNET model generated. See Black Console for
model details\n")
    '''
    model.compile(optimizer=Adam(lr=1e-5), loss=dice_coef_loss,
metrics=[dice_coef, 'binary_accuracy'])
```

```
print(model.summary())
    model.compile(optimizer=Adam(lr=2e-4), loss=[dice_coef_loss],
metrics = [dice_coef, 'binary_accuracy'])
    train_vol, validation_vol, train_seg, validation_seg =
train_test_split((images-127.0)/127.0,

(mask>127).astype(np.float32),
                                                        test_size =
0.1,random_state = 2018)
    train_vol, test_vol, train_seg, test_seg =
train_test_split(train_vol,train_seg,
                                                        test_size =
0.1,

random_state = 2018)
    hist = model.fit(x = train_vol, y = train_seg, batch_size = 16,
epochs = 50, validation_data =(test_vol,test_seg))
    model.save_weights('model/model_weights.h5')
    model_json = model.to_json()
    with open("model/model.json", "w") as json_file:
        json_file.write(model_json)
    f = open('model/history.pckl', 'wb')
    pickle.dump(hist.history, f)
    f.close()
    '''


def getSegmentation():
    img = cv2.imread('myimg.png')
    orig = cv2.imread('test1.png')
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    thresh = cv2.threshold(gray, 30, 255, cv2.THRESH_BINARY)[1]
    contours = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    contours = contours[0] if len(contours) == 2 else contours[1]
    min_area = 0.95*180*35
    max_area = 1.05*180*35
    result = orig.copy()
    for c in contours:
        area = cv2.contourArea(c)
        cv2.drawContours(result, [c], -1, (0, 0, 255), 10)
        if area > min_area and area < max_area:
            cv2.drawContours(result, [c], -1, (0, 255, 255), 10)
    return result




ydef TumourSegmentation():

global model
    filename = filedialog.askdirectory(initialdir="testSamples")
    img = cv2.imread(str(filename)+'/t2.png',0)
    img = cv2.resize(img,(64,64), interpolation = cv2.INTER_CUBIC)
    img = img.reshape(1,64,64,1)
    img = (img-127.0)/127.0
    preds = model.predict(img)
```

```
    preds = preds[0]
    print(preds.shape)
    orig = cv2.imread(str(filename)+'/t2.png',0)
    orig = cv2.resize(orig,(300,300),interpolation = cv2.INTER_CUBIC)
    cv2.imwrite("test1.png",orig)

    flair = cv2.imread(str(filename)+'/flair.png',0)
    flair = cv2.resize(flair,(300,300),interpolation = cv2.INTER_CUBIC)
    t1 = cv2.imread(str(filename)+'/t1.png',0)
    t1 = cv2.resize(t1,(300,300),interpolation = cv2.INTER_CUBIC)
    t1ce = cv2.imread(str(filename)+'/t1ce.png',0)
    t1ce = cv2.resize(t1ce,(300,300),interpolation = cv2.INTER_CUBIC)

    preds = cv2.resize(preds,(300,300),interpolation = cv2.INTER_CUBIC)
    cv2.imwrite("myimg.png",preds*255)
    preds = getSegmentation()
    cv2.imshow('Flair Image',flair)
    cv2.imshow('T1',t1)
    cv2.imshow("T1ce Image",t1ce)
    cv2.imshow('T2 Image',orig)
    cv2.imshow("Label Image",preds)
    cv2.waitKey(0)

def graph():
    f = open('model/history.pckl', 'rb')
    data = pickle.load(f)
    f.close()
    dice = data['dice_coef']
    for i in range(len(dice)):
        dice[i] = dice[i] * 2
    plt.figure(figsize=(10,6))
    plt.grid(True)
    plt.xlabel('Iterations')
    plt.ylabel('Dice Score')
    plt.plot(dice, 'ro-', color = 'green')
    plt.legend(['Dice Score'], loc='upper left')
    #plt.xticks(wordloss.index)
    plt.title('Iteration Wise Dice Score Graph')
    plt.show()



font = ('times', 16, 'bold')
title = Label(gui, text='Brain Tumour Image Segmentation Using Deep
Networks')
title.config(bg='LightGoldenrod1', fg='medium orchid')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)


font1 = ('times', 12, 'bold')
text=Text(gui,height=20,width=100)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=300)
text.config(font=font1)
```

```
font1 = ('times', 12, 'bold')
loadButton = Button(gui, text="Upload BRATS Dataset",
command=uploadDataset)
loadButton.place(x=50,y=100)
loadButton.config(font=font1)


uploadButton = Button(gui, text="Generate CNN & UNET Model",
command=generateModel)
uploadButton.place(x=50,y=150)
uploadButton.config(font=font1)


descButton = Button(gui, text="Upload Test Image & Segmentation",
command=TumourSegmentation)
descButton.place(x=50,y=200)
descButton.config(font=font1)

closeButton = Button(gui, text="Dice Similarity Graph", command=graph)
closeButton.place(x=50,y=250)
closeButton.config(font=font1)



gui.config(bg='OliveDrab2')

gui.mainloop()
```

8. SOFTWARE TESTING

8.1 Implementation

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification.

The computer system and its environment are tested to the satisfaction of the user.The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly.

The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

8.2 Types of testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition.

Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

8.2.1 System Testing

Testing has become an System integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical.

When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

8.2.2 Module Testing

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result.

In the module testing phase of brain tumor segmentation using CNN and UNet models, the trained networks are deployed to segment unseen or test MRI images.

Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately.

8.2.3 Integration Testing

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct.

In the context of brain tumor segmentation using CNN and UNet models, integration testing plays a crucial role in ensuring the seamless interaction and functionality of various components within the segmentation system.

Integration testing involves evaluating how individual modules or components, such as data preprocessing, model architecture, training procedures, and inference pipelines, integrate and collaborate with each other to achieve the desired segmentation outcome.

8.2.4 Acceptance Testing

When that user fined no major problems with its accuracy the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management.

9. SYSTEM DESIGN

9.1 UML DIAGRAMS

9.1.1 Class Diagram

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

- The upper part holds the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake.

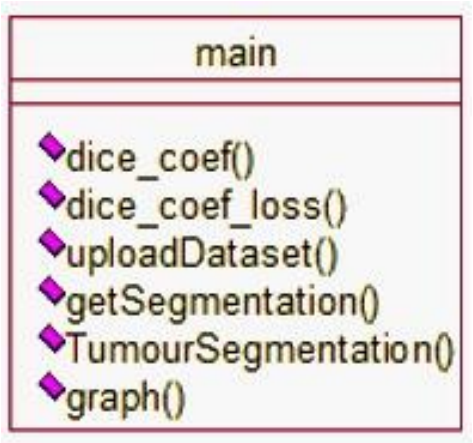

Fig.9.1.1 Class diagram

9.1.2 Use case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

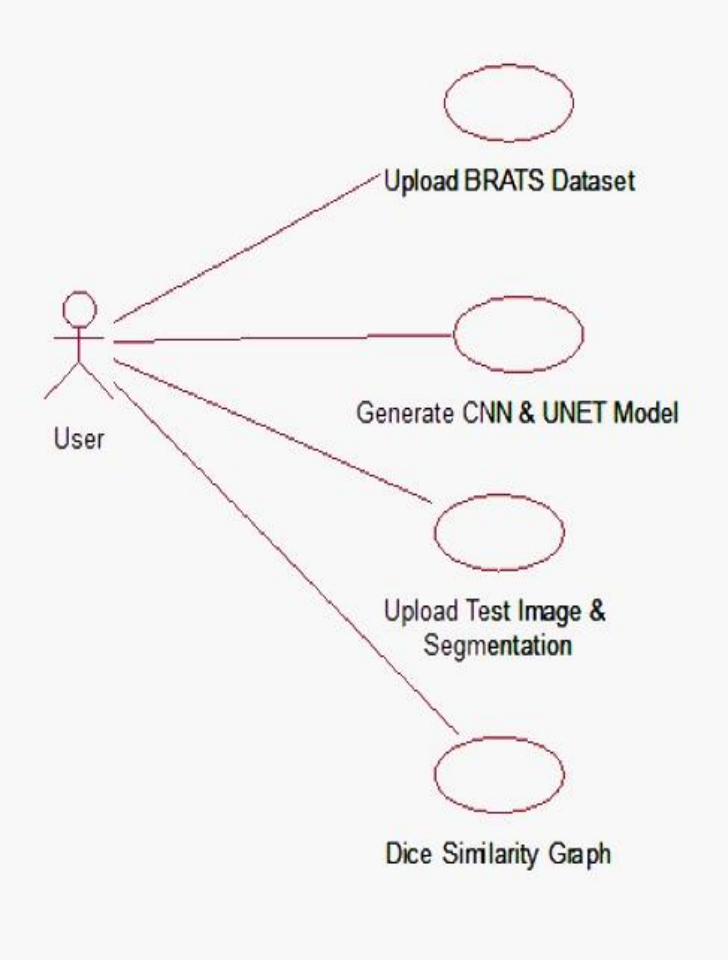

Fig.9.1.2 Use case diagram

9.1.3 Sequence Diagram

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

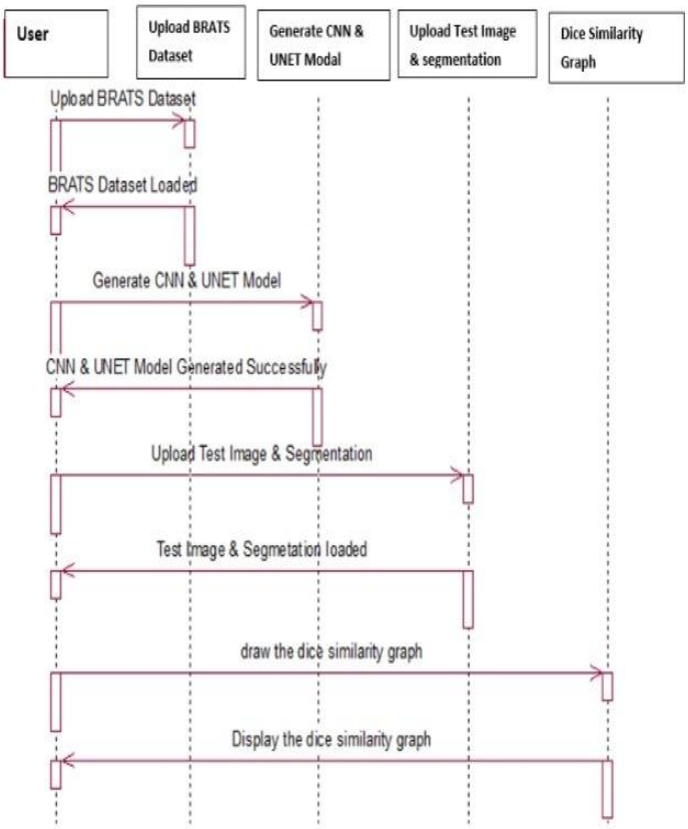

Fig.9.1.3 Sequence Diagram

9.1.4 Component Diagram

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

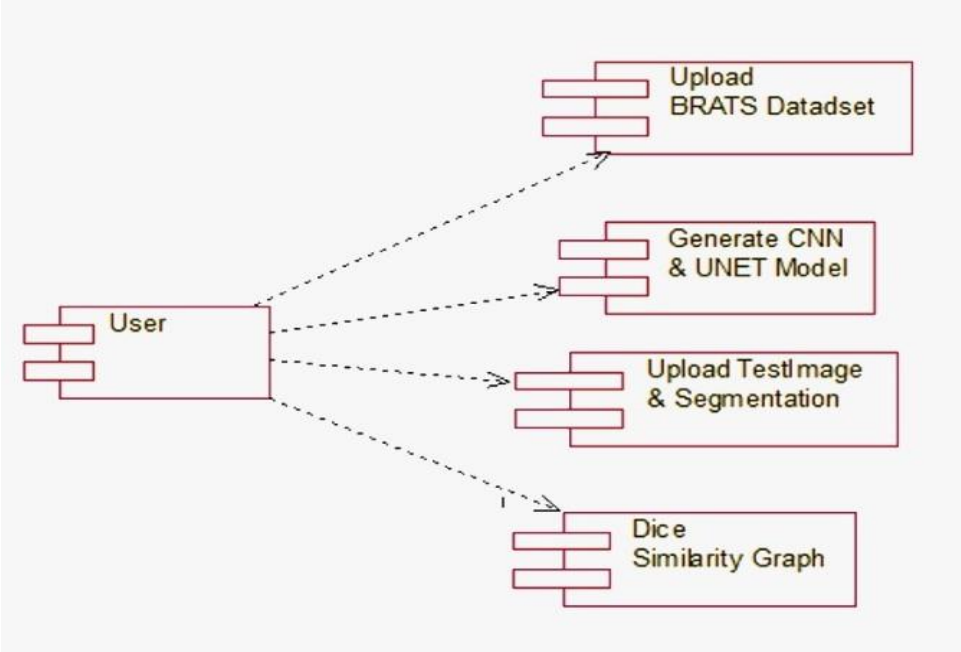

Fig.9.1.4 Component diagram

9.1.5 Deployment Diagram

A deployment diagram in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

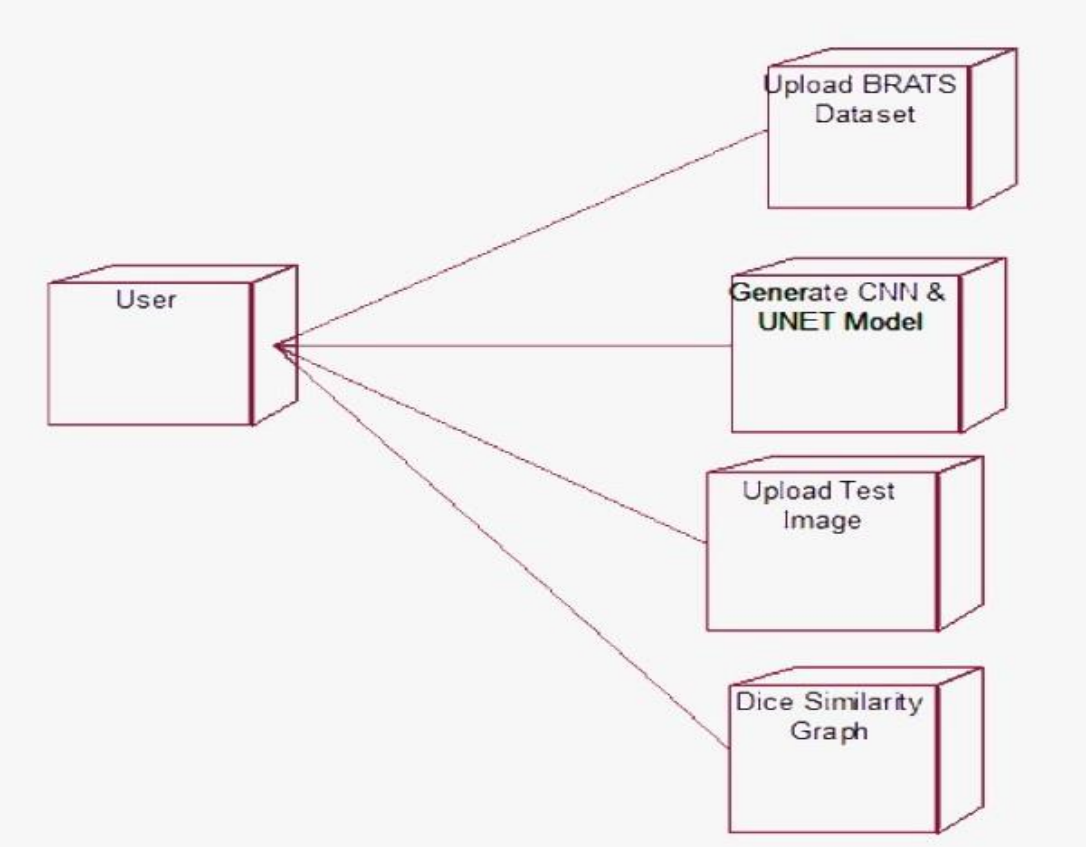

Fig.9.1.5 Deployment diagram

9.1.6 Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

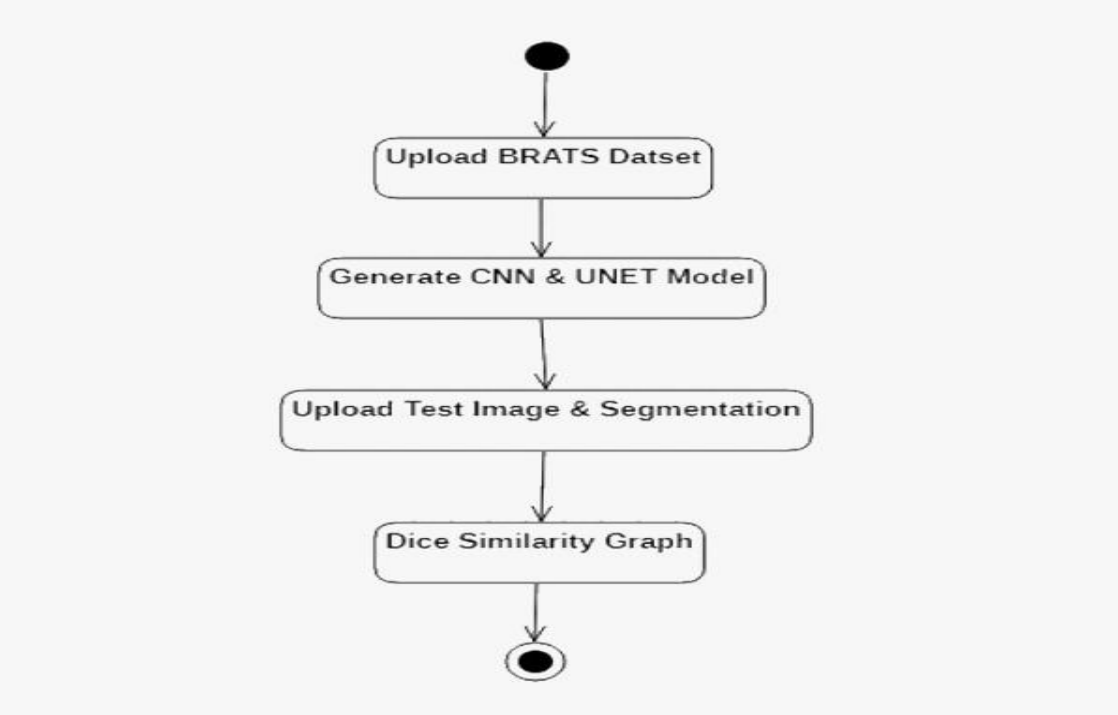

Fig.9.1.6 Activity diagram

9.1.7 Data Flow Diagram

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs.

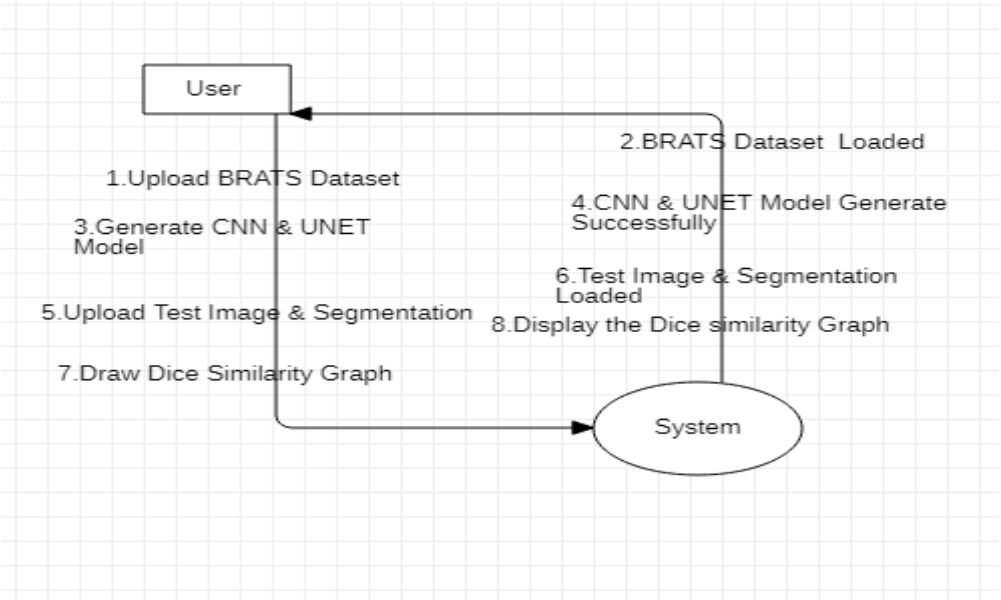

Fig.9.1.7 Data Flow diagram

10. OUTPUT SCREENSHOTS

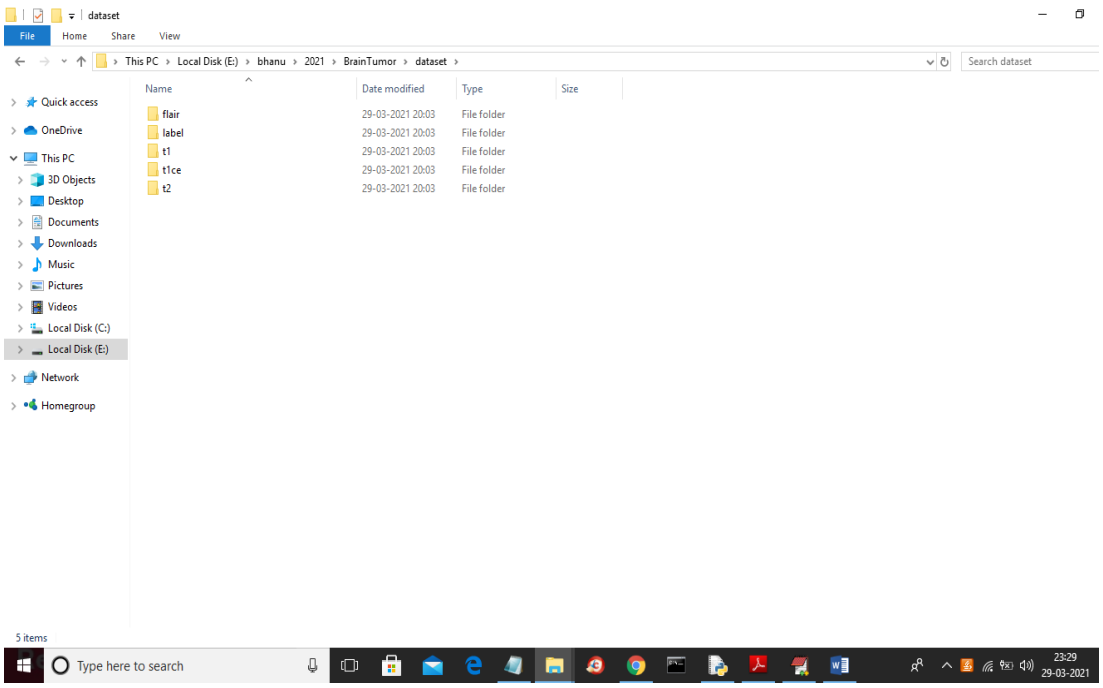

Fig.10.1 Dataset Content

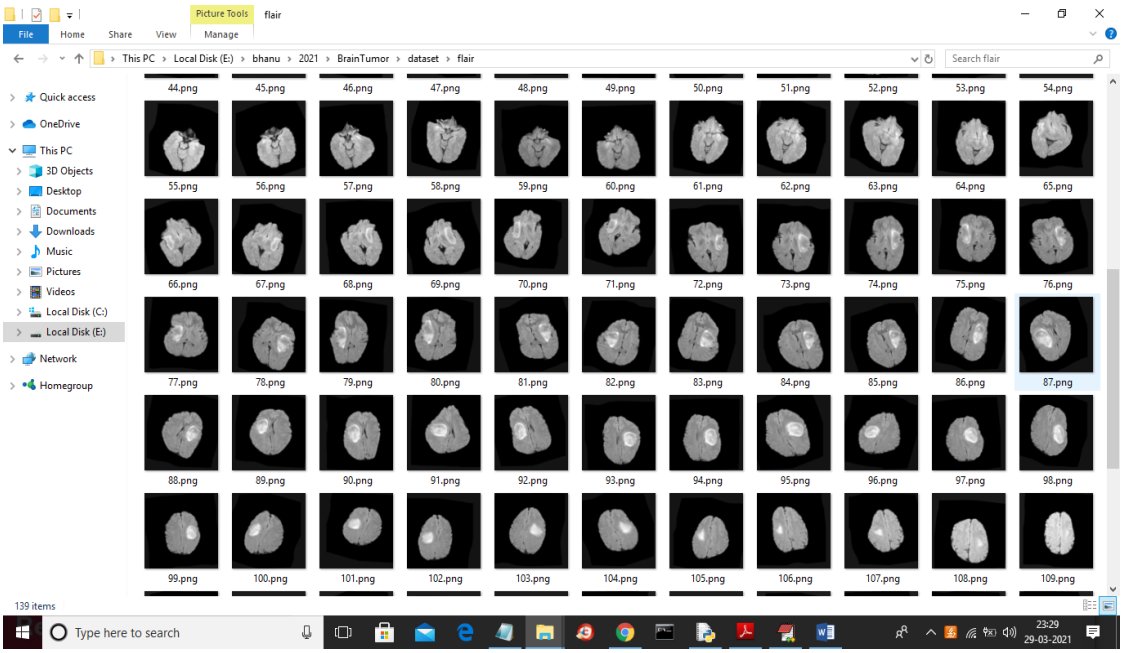

Fig.10.2 Dataset is used to train CNN and UNET model

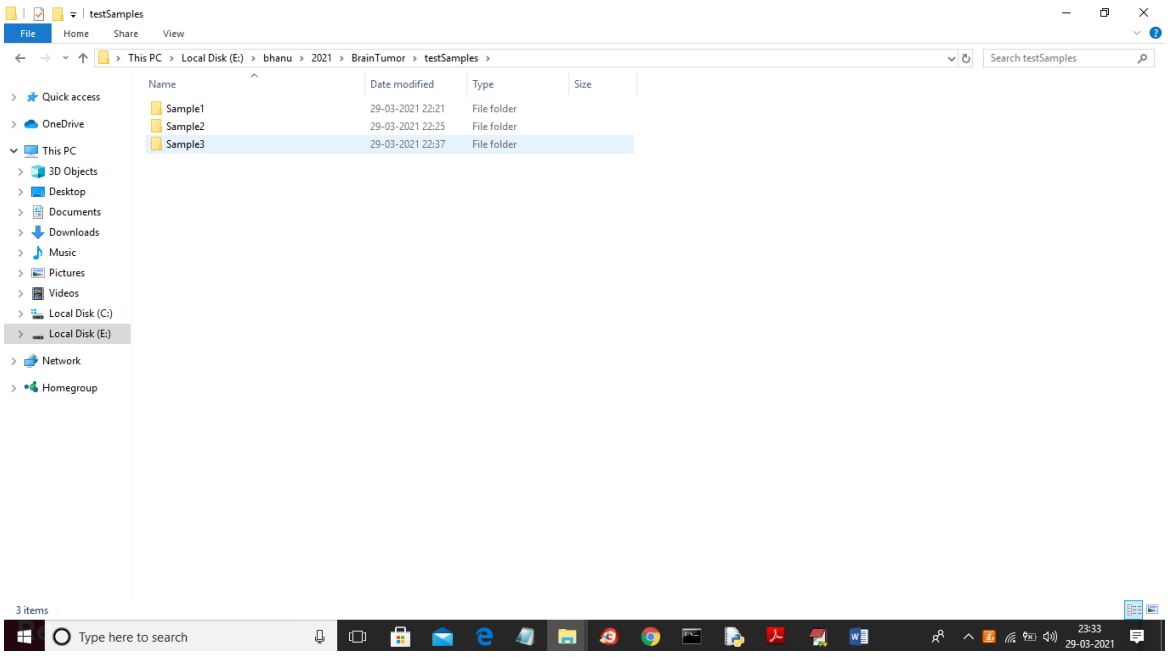

Fig.10.3 Samples images

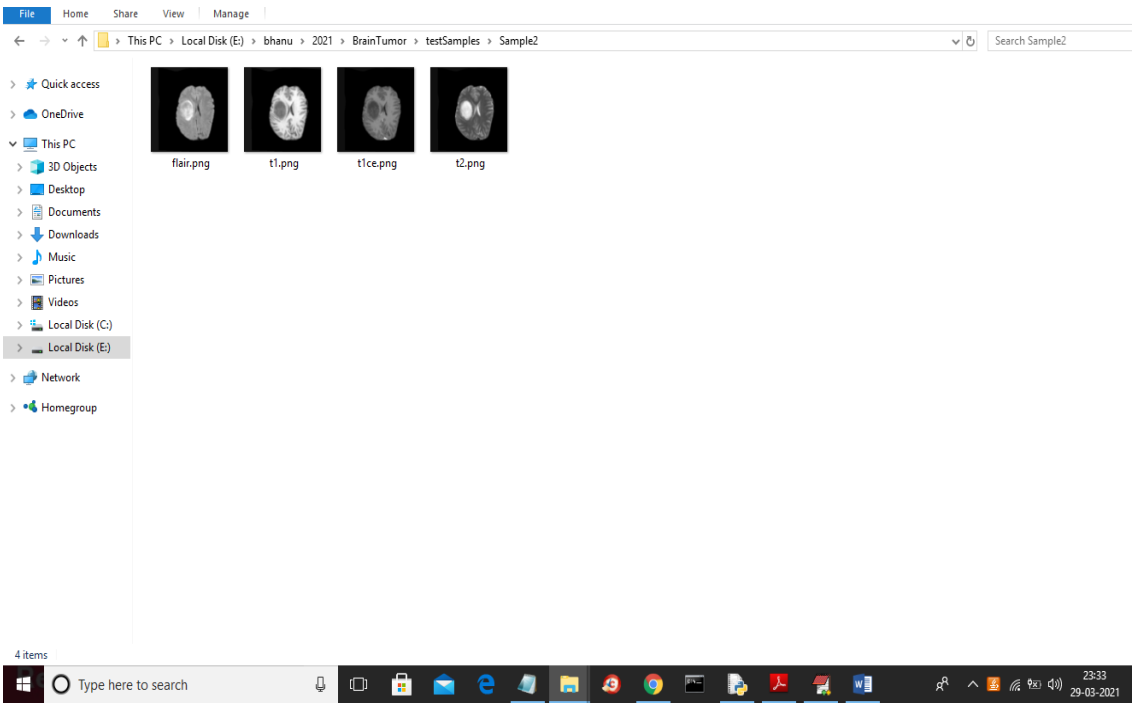

Fig.10.4 Segmented label image

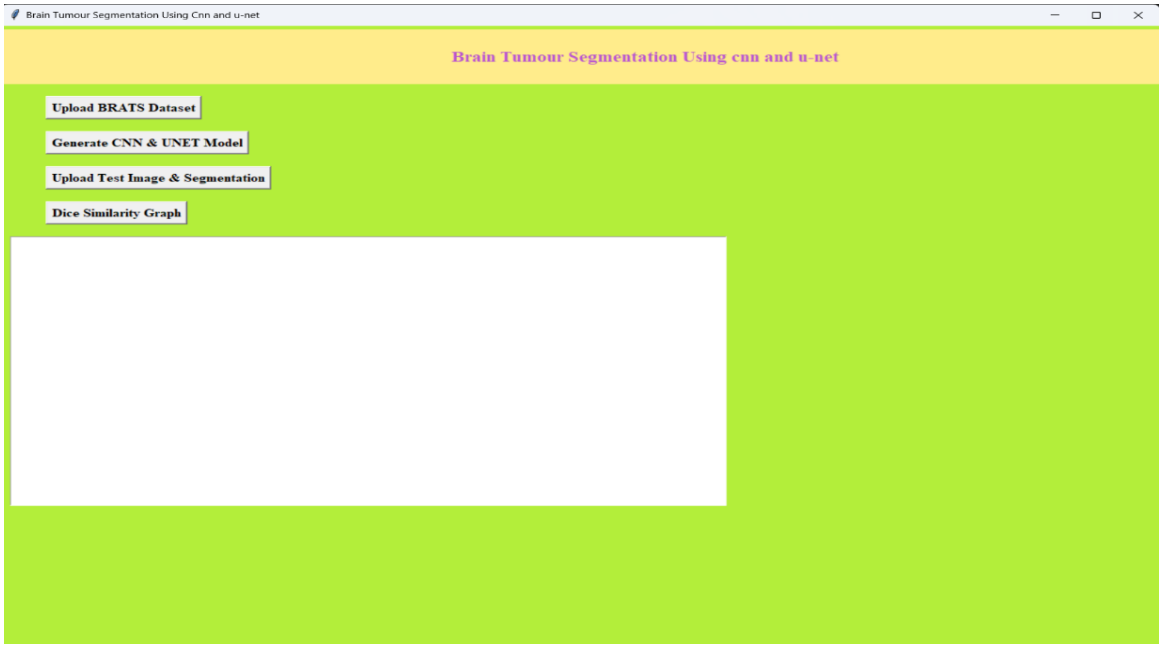

Fig.10.5 Upload BRATS Dataset

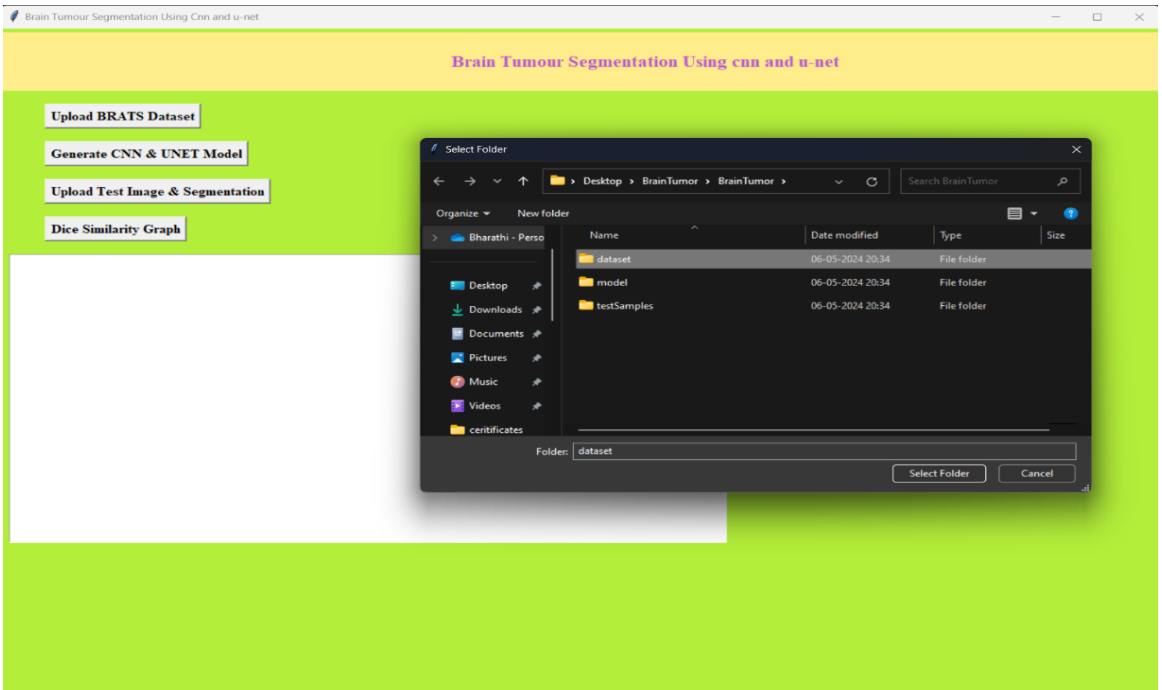

Fig.10.6 Uploading 'dataset' folder

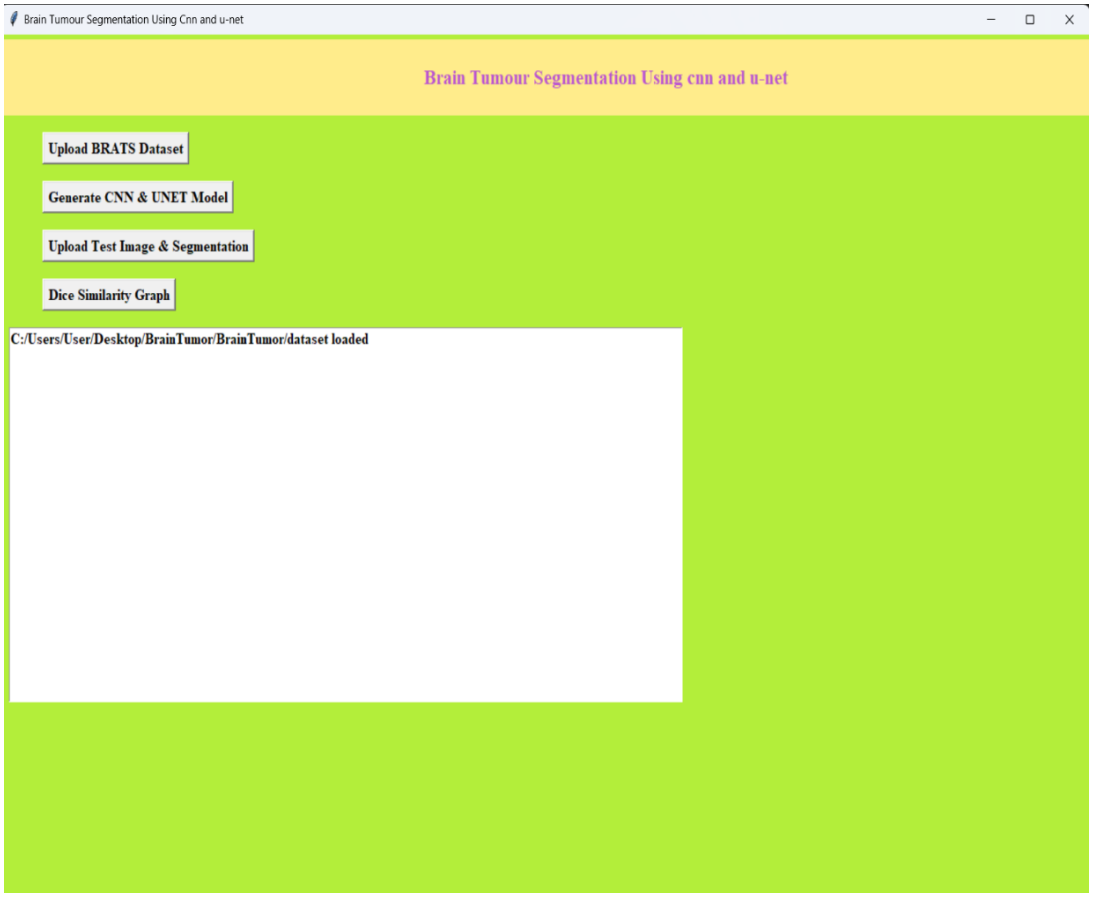

Fig.10.7 Dataset loaded

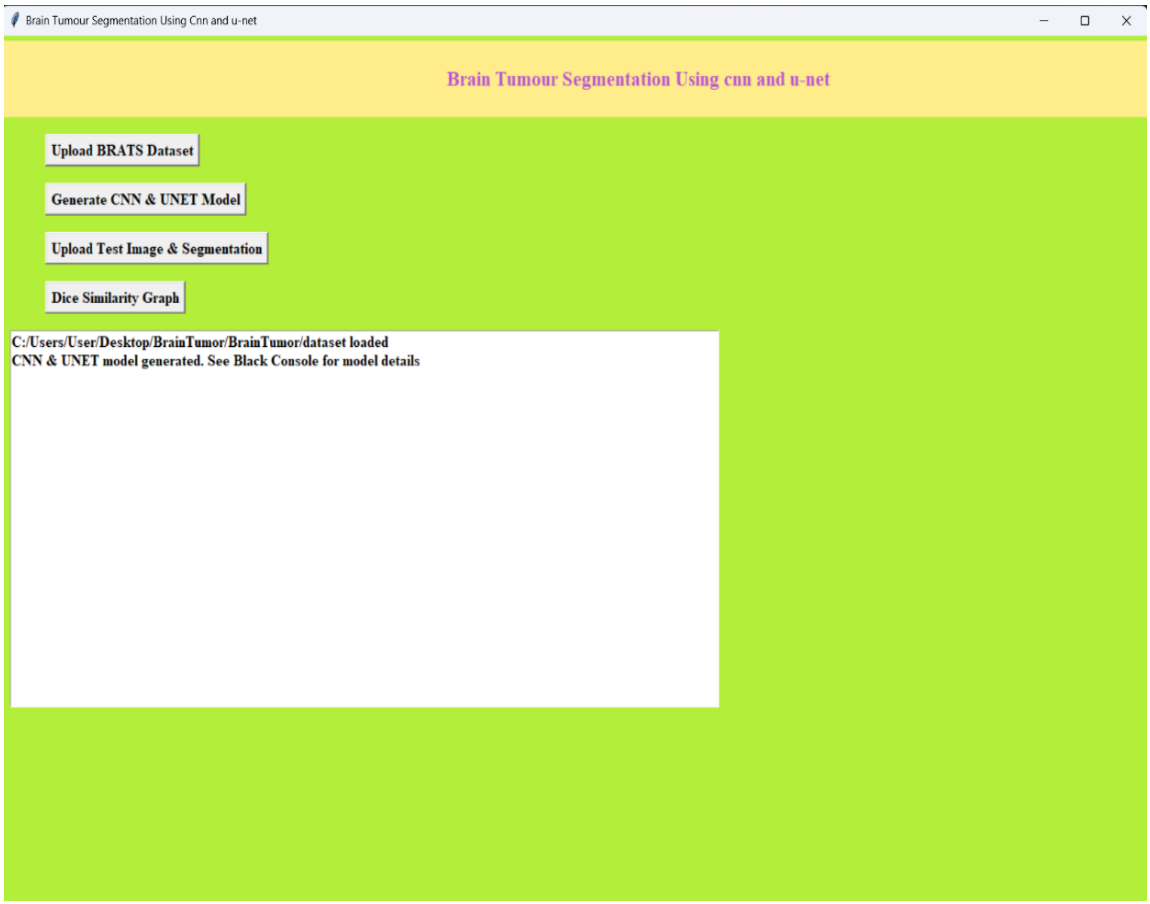

Fig.10.8 Generated CNN & UNET Model

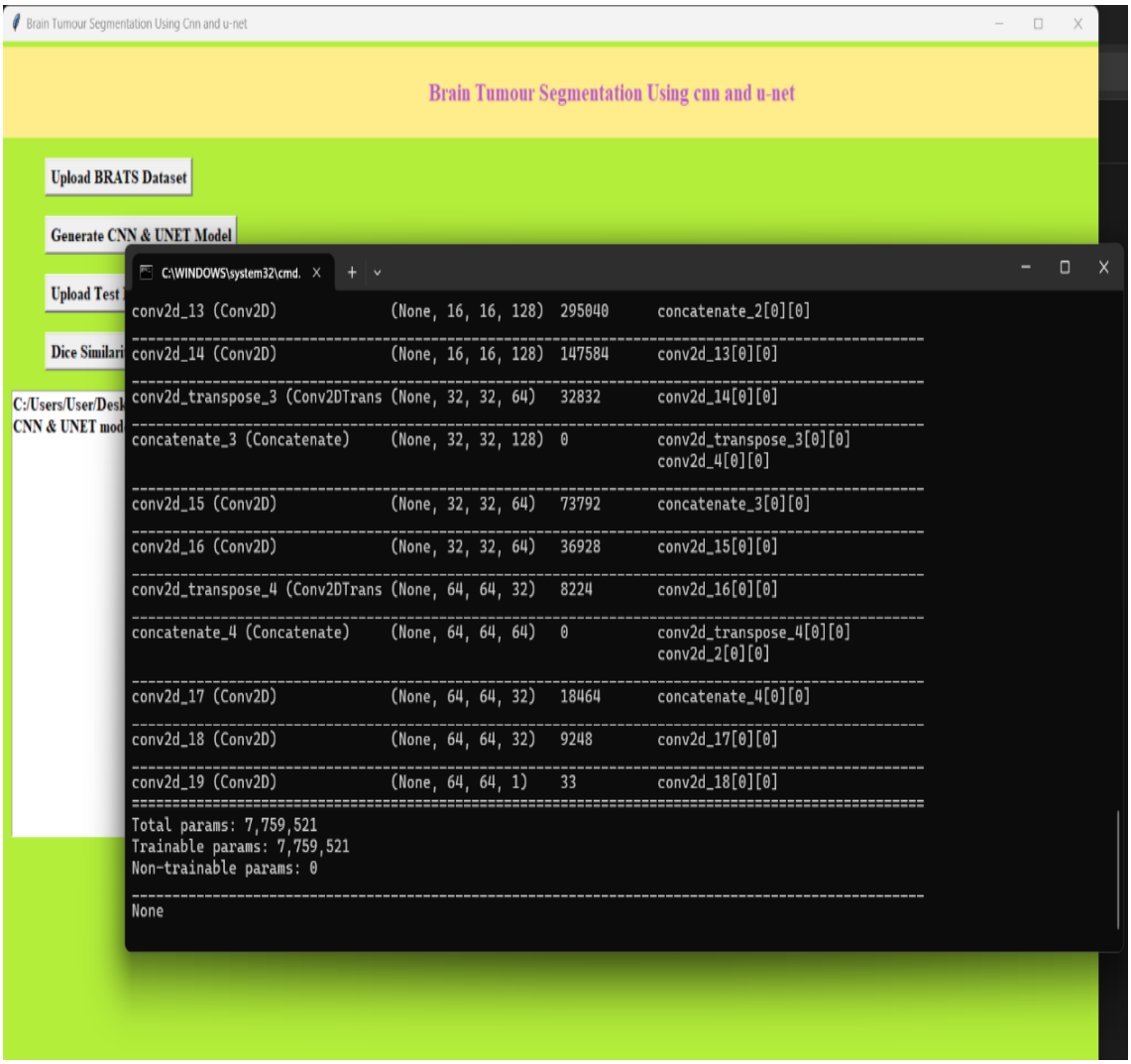

Fig.10.9 Console to see CNN and UNET layer details

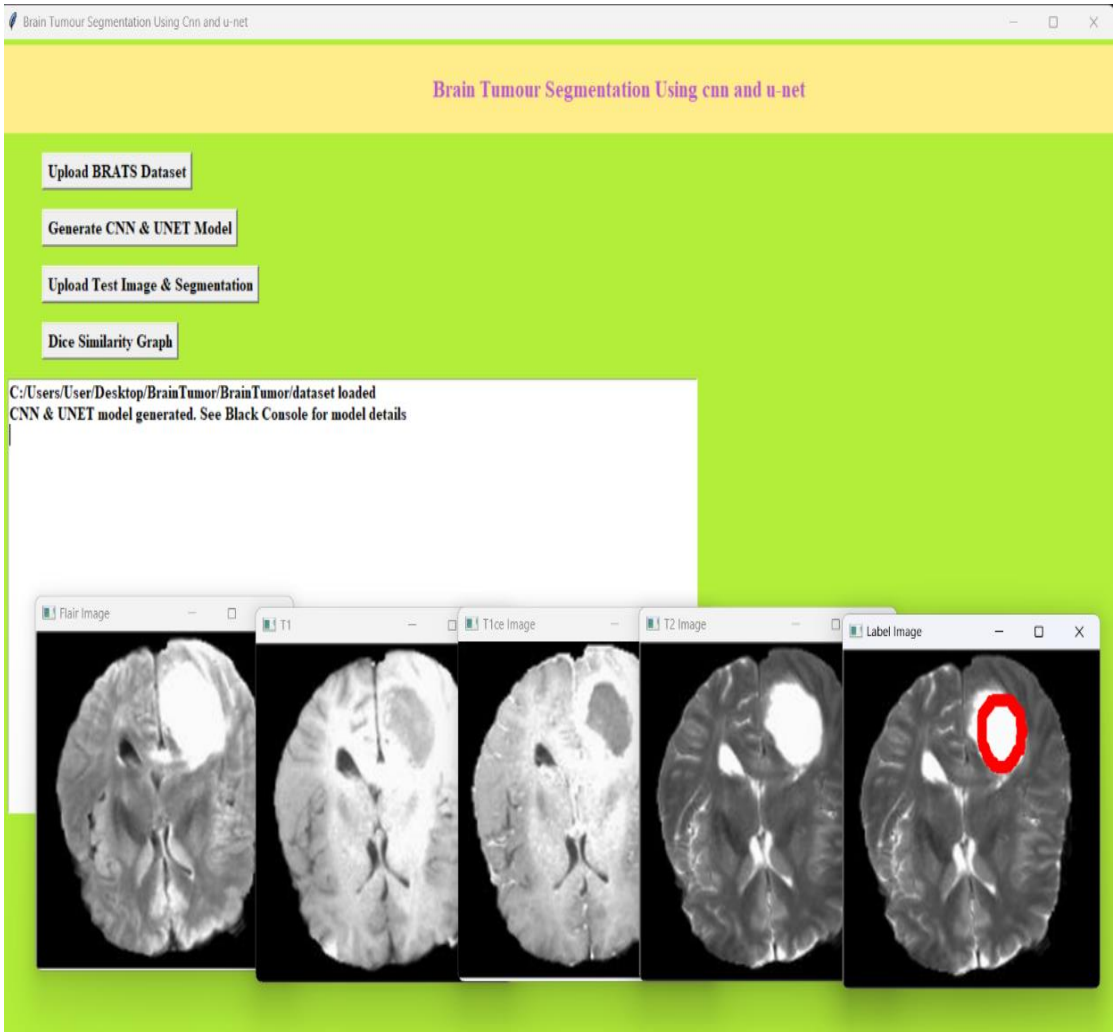

Fig.10.10 Predicted image with segmented part

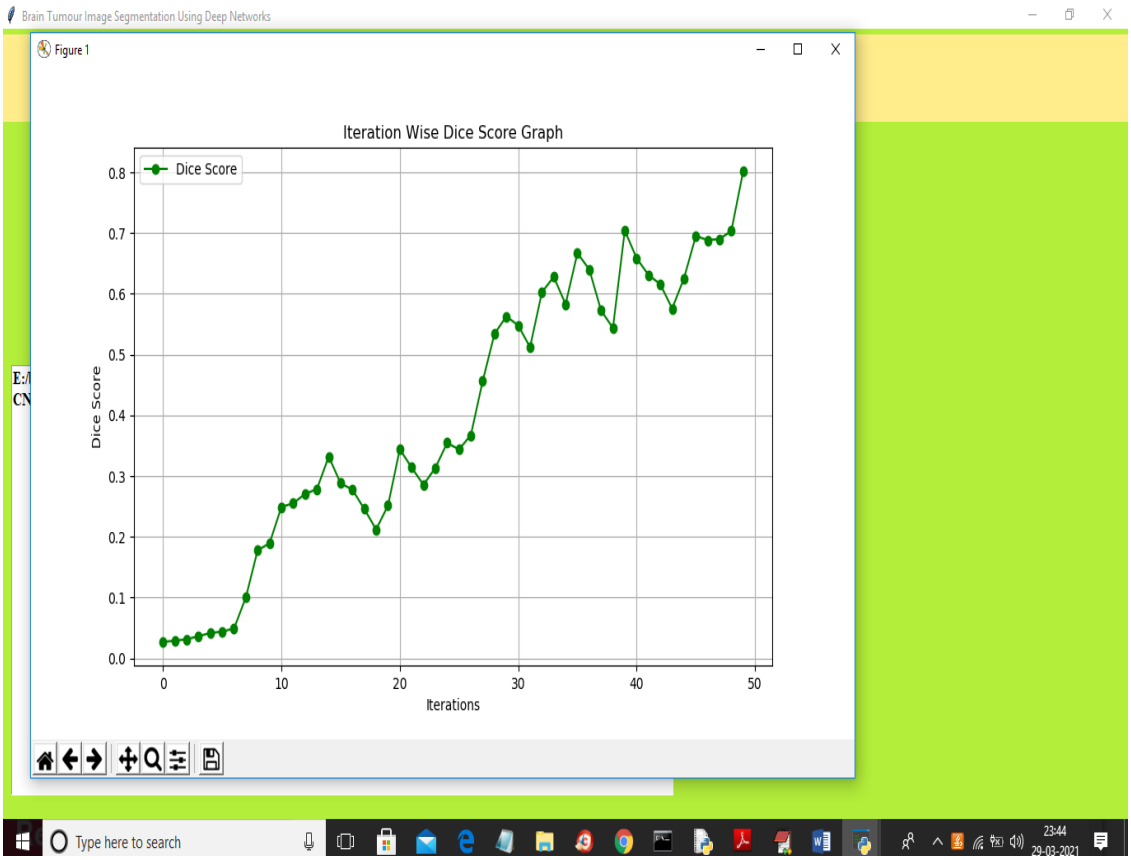

Fig.10.11 Dice score representation

11. CONCLUSION AND FUTURE SCOPE

11.1 CONCLUSION

In this work, we have described an ensemble of two networks, both of which are individually used frequently on the task of biomedical image segmentation. The ensemble successfully generates highly accurate segmentation of brain tumours from the multimodal MRI scans as provided by the BraTS 2019 challenge, which compares favourably with predictions given from various other state of the art models. We use a method of variable ensembling to combine the respective outputs from the model to achieve the best scores. The proposed ensemble offers an automated and objective method of generating brain tumour segmentation to aid in disease planning and patient management clinically.

11.2 FUTURE SCOPE

It is observed on extermination that the proposed approach needs a vast training set for better accurate results; in the field of medical image processing, the gathering of medical data is a tedious job, and, in few cases, the datasets might not be available. In all such cases, the proposed algorithm must be robust enough for accurate recognition of tumor regions from MR Images. The proposed approach can be further improvised through in cooperating weakly trained algorithms that can identify the abnormalities with a minimum training data and also self-learning algorithms would aid in enhancing the accuracy of the algorithm and reduce the computational time.

REFERENCES

[1] S. Bauer, R. Wiest, L. P. Nolte, and M. Reyes, "A survey of MRI-based medical image analysis for brain tumour studies," 2013, [Online]. Available: http://www.

[2] R. Leece, J. Xu, Q. T. Ostrom, Y. Chen, C. Kruchko, and J. S. Barnholtz-Sloan, "Global incidence of malignant brain and other central nervous system tumours by histology, 2003--2007," *Neuro. Oncol.*, vol. 19, no. 11, pp. 1553–1564, 2017.

[3] T. A. Dolecek, J. M. Propp, N. E. Stroup, and C. Kruchko, "CBTRUS statistical report: primary brain and central nervous system tumours diagnosed in the United States in 2005--2009," *Neuro. Oncol.*, vol. 14, no. suppl_5, pp. v1--v49, 2012.

[4] D. N. Louis *et al.*, "The 2016 World Health Organization classification of tumours of the central nervous system: a summary," *Acta Neuropathol.*, vol. 131, no. 6, pp. 803–820, 2016.

[5] R. Stupp *et al.*, "Radiotherapy plus concomitant and adjuvant temozolomide for glioblastoma," *N. Engl. J. Med.*, vol. 352, no. 10, pp. 987–996, 2005.

[6] S. Bakas *et al.*, "Identifying the best machine learning algorithms for brain tumour segmentation, progression assessment, and overall survival prediction in the BRATS challenge," *arXiv Prepr. arXiv1811.02629*, 2018.

[7] B. H. Menze, K. Van Leemput, D. Lashkari, M.-A. Weber, N. Ayache, and P. Golland, "A generative model for brain tumour segmentation in multimodal images," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2010, pp. 151–159.

[8] S. Bakas *et al.*, "Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features," *Sci. data*, vol. 4, p. 170117, 2017.

[9] S. Bakas *et al.*, "Segmentation labels and radiomic features for the pre-operative scans of the TCGA-LGG collection," *cancer imaging Arch.*, vol. 286, 2017.

[10] S. Bakas *et al.*, "Segmentation labels and radiomic features for the pre-operative scans of the TCGA-GBM collection. The Cancer Imaging Archive," *Nat Sci Data*, vol. 4, p. 170117, 2017.