A

MAJOR PROJECT REPORT

ON

Brain Tumor Detection and Classification using MRI Scan Images

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Submitted By

Thakkalapally Sai Sushanth	**245318733110**
Sarvaraju Sai Karthik	**245318733104**

Under the guidance

Of

Mrs.M.Meenakshi

Assistant Professor

Department of Computer Science & Engineering

NEIL GOGTE INSTITUTE OF TECHNOLOGY

NEIL GOGTE INSTITUTE OF TECHNOLOGY

A Unit of Keshav Memorial Technical Education (KMTES)
Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

CERTIFICATE

*This is to certify that the project work entitled "***Brain Tumor Detection and Classification using MRI Scan Images"** *is a bonafide work carried out by* **Thakkalapally Sai Sushanth (245318733110), Sarvaraju Sai Karthik (245318733104)** of IV year VIII semester **Bachelor of Engineering** *in* **COMPUTER SCIENCE & ENGINEERING** *by Osmania University, Hyderabad during the academic year* **2018-2022** *is a record of bonafide work carried out by them. The results embodied in this report have not been submitted to any other University or Institution for the award of any degree*

Internal Guide	**Head of Department**
Mrs.M.Meenakshi	Dr.K.V.Ranga Rao
(Assistant Professor)	(HOD)

External

NEIL GOGTE INSTITUTE OF TECHNOLOGY
A Unit of Keshav Memorial Technical Education (KMTES)
Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

DECLARATION

We hereby declare that the Major Project Report entitled, "**Brain Tumor Detection and Classification using MRI Scan Images** "submitted for the B.E degree is entirely our work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree.

Date: 26th May, 2022

Thakkalapally Sai Sushanth (245318733110)

Sarvaraju Sai Karthik (245318733104)

ACKNOWLEDGEMENT

We are happy to express our deep sense of gratitude to the principal of the college **Dr. D Jaya Prakash, Professor**, Neil Gogte Institute of Technology, for having provided us with adequate facilities to pursue our project.

We would like to thank**, Dr.K.V.Ranga Rao**, **Head of the Department,** Computer Science & Engineering, Neil Gogte Institute of Technology, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

We would also like to thank my internal guide **Mrs.M.Meenakshi, Assistant Professor** for her Technical guidance & constant encouragement.

We sincerely thank our seniors and all the teaching and non-teaching staff of the Department of Computer Science & Engineering and Information Technology for their timely suggestions, healthy criticism and motivation during the course of this work.

Finally, we express our immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at the right time for the development and success of this work.

I

ABSTRACT

A Brain tumor is considered as one of the most aggressive diseases and its segmentation is most crucial tasks in the terrain of medical image processing. Human-assisted manual classification for prediction and diagnosis of Tumor is inaccurate. Proper treatment, planning, and accurate diagnostics should be implemented to improve the life expectancy of the patients.

The best technique to detect brain tumors is Magnetic Resonance Imaging (MRI). A huge amount of image data is generated through the scans and this manual examination can be error-prone due to the level of complexities involved in brain tumors.

In this project, Convolution Neural Network (CNN) is applied in detecting the presence of brain tumor and their performance is analysed. It distinguishes between normal and abnormal pixels, based on texture and statistical features. This improves the performance, minimises the complexity and works on real time data.

II
TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
	ACKNOWLEDGEMENT	**I**
	ABSTRACT	**II**
	LIST OF FIGURES	**V**
	LIST OF TABLES	**V**
	INTRODUCTION	**9-11**
	1.1 PROBLEM STATEMENT	**10**
	1.2 MOTIVATION	**11**
	1.3 OUTLINE	**11**
2.	**LITERATURE SURVEY**	**12-14**
	2.1 EXISTING SYSTEM	**12**
	2.2 PROPOSED SYSTEM	**12**
	2.3 RELATED WORK	**12-14**
3.	**SOFTWARE REQUIREMENTS SPECIFICATION**	**15-16**
	3.1 OVERALL DESCRIPTION	**15**
	3.2 OPERATING ENVIRONMENT	**16**
	3.3 FUNCTIONAL REQUIREMENTS	**16**
	3.4 NON-FUNCTIONAL REQUIREMENTS	**16**
4.	**DESIGN DIAGRAMS**	**17-21**
	4.1 SYSTEM ARCHITECTURE	**17**
	4.2 USE-CASE DIAGRAM	**18**
	4.3 CLASS DIAGRAM	**19**

4.4 SEQUENCE DIAGRAM **20**

V

5. IMPLEMENTATION

5.1 SAMPLE CODE **22-36**

6. TESTING **37-38**

6.1 TEST CASES **38**

7. SCREENSHOTS **39-41**

8. CONCLUSION AND FUTURE SCOPE **42**

9 BIBLIOGRAPHY & REFERENCES **43**

List of Figures

Figure No.	Name of Figure	Page No.
1.	Use case Diagram	18
2.	Class Diagram	19
3.	Sequence Diagram	20
4	Activity Diagram	21

List of Tables

Table No.	Name of Table	Page No.
1.	Testcase #1	38

1. INTRODUCTION

Medical imaging refers to several techniques that can be used as non-invasive methods of looking inside the body . The main use of medical image in the human body is for treatment and diagnostic purposes. So, it plays a significant role in the betterment of treatment and the health of the human.

Image segmentation is a crucial and essential step in image processing that determines the success of image processing at a higher level . In this case we have mainly focused on the segmentation of the brain tumour from the MRI images. It helps the medical representatives to find the location of the tumour in the brain easily. Medical image processing encompasses the utilization and exploration of 3D image datasets of the physical body, obtained most typically from computed tomography (CT) or Magnetic Resonance Imaging (MRI) scanner to diagnose pathologies or guide medical interventions like surgical planning, or for research purposes. Medical image processing is applied by radiologists, engineers, and clinicians to understand the anatomy of either individual patients or population groups highly. Measurement, statistical analysis, and creation of simulation models which incorporate real anatomical geometries provide the chance for more complete understanding, as an example of interactions between patient anatomy and medical devices.

Tumour: The word “Tumour” is a synonym for the word “neoplasm” which is formed by an abnormal growth of cells. A tumour is significantly different from cancer .

Classification of tumour

There are three basic types of tumours: 1) Benign; 2) Pre-Malignant; 3) Malignant (cancer can only be malignant).

Benign tumour

A Benign Tumour is not always Malignant or cancerous. It might not invade close tissue or unfold to alternative components of the body the way cancer can. In most cases, the outlook with benign tumours is not at all serious but it can be serious if it presses on vital structures such as blood vessels or nerves.

Pre-Malignant tumour

In these tumours, the cells are not cancerous. However, they need the potential to become malignant. The cells will grow and unfold to alternative components of the body.

Malignant tumour

Malignancy (mal- = “bad” and ignis = “fire”) Malignant tumours area are cancerous. They develop once cells grow uncontrollably. If the cells still grow and unfold, the malady will become dangerous. Malignant tumours will grow quickly and unfold to alternative components of the body during a method known as metastasis.

In this project we are exploring Malignant brain tumours and it’s types. They are mainly Glioma, Meningioma, pituitary tomors.

1.1 PROBLEM STATEMENT

- Possibility of human error.
- Misdiagnosis due to ambiguity of location.
- In countries like India where doctor to patient ratio is very low it makes it hard for patient and doctors to spare enough time on examining and diagnosing.

1.2 MOTIVATION

The main motivation behind Brain tumor detection is to not only detect tumor but it can also classify types of tumor. So it can be useful in cases such as we have to sure the tumor is positive or negative, it can detect tumor from image and return the result tumor is positive or not. This project deals with such a system, which uses computer, based procedures to detect tumor blocks and classify the type of tumor using Convolution Neural Network Algorithm for MRI images of different patients.

1.3 OUTLINE

The main objective is to provide doctors good software to identify and classify tumors.These are detected by Magnetic Resonance Imaging (MRI). In this project, Convolution Neural Network (CNN) is applied in detecting the presence of brain tumor and their performance is analysed. This distinguishes between normal and abnormal pixels, based on texture and statistical features. This improves the performance, minimises the complexity and works on real time data.

2. LITERATURE SURVEY

2.1 EXISTING SYSTEM

In existing system, the tumour is manually inspected by the radiologists and this examination can be error-prone due to the level of complexities involved in brain tumours and their properties.

As it is a Human written report, there is a possibility of errors.

2.2 PROPOSED SYSTEM

We propose a system performing detection and classification by using Deep Learning Algorithms using Convolution Neural Network (CNN), Artificial Neural Network (ANN), and Transfer Learning (TL) would be helpful to doctors all around the world.

2.3. RELATED WORK

Brain Tumour segmentation methods can be divided as three parts. Manual methods, Semi-automatic methods and Absolute automatic methods. We can determine it according to the level of user interaction required .

2.3.1. MANUAL SEGMENTATION METHODS

It needs a medical specialist to use the different information picturize by the MRI images along with anatomical and physiological knowledge achieve through training and experience. This procedure requires the medical specialist going through multiple slices of images part by part, analyzing the brain Tumour and manually cropping the tumour regions carefully. It's a time consuming task as manual segmentation is also doctor dependent and segmentation results are subject to large intra and inter ratter variability [7]. Although, this is widely applied to execute the results of semi-automatic and fully automatic techniques.

2.3.2. SEMI-AUTOMATIC SEGMENTATION METHODS

It needs the reaction of the user for three main purposes; initialization, intervention or feedback response and evaluation . Initialization is mainly executed by defining a region of interest (ROI), restraining the estimated Tumour area, for the automatic algorithm to process. Parameters of pre-processing technique can also be balanced to fit the input images. In addition to initialization, automated algorithms can be directed towards a necessary result throughout the procedure by receiving feedbacks. This process also provides the adjustments in response. Again, user can estimate the results and change or repeat the procedure again if not satisfied. Hamamci et al. proposed the "Tumour Cut" method . This method comprised applying the algorithm separately to each MRI modality (e.g. T1, T2, T1-Gd and FLAIR). Then we combine the outcome to obtain the final tumour volume. A current semi-automatic method applied to a novel classification approach . In this technique segmentation problem was converted into a classification problem and a brain tumour is segmented by training and classifying within that same brain only. Commonly, a machine learning classification technique, for brain tumour segmentation, needs a large quantity of brain MRI scans images (with checked answers) from different cases to train. This outcome in a necessity handles intensity bias correction and other noises. Although in this approach, user initializes the procedure by sort out a subset of voxels linked with each tissue type, from a single case. For these subsets of voxels, algorithm extracts the intensity values along with spatial coordinates as features and trains a support vector machine (SVM) that is used to classify all the voxels of the same image to their corresponding tissue type. Semi-automatic brain tumour segmentation approach not only takes reduces time than manual method but also it can maintain efficient results but still prone to intra and inter-rater user variability. Therefore, recent brain tumour segmentation research is mainly focused on fully automatic methods.

2.3.3. ABSOLUTE AUTOMATIC SEGMENTATION METHODS

In this approach user does not need any interaction. Most importantly, artificial intelligence and preparatory knowledge are merged to solve the segmentation problem.

2.3.3.1. CHALLENGES

Automatic segmentation of gliomas is a very tuff and important problem. Brain tumour MRI data obtained from clinical scans or synthetic databases are naturally complicated. The devices for MRI and protocols that are using for acquisition can vary significantly from scan to scan imposing intensity biases and other variations for each different part of image in the dataset. Several modalities need to significantly segment tumour sub-regions even adds to this complexity.

3. SOFTWARE REQUIREMENTS

3.1 OVERALL DESCRIPTION

This SRS is an outline of the entire task situation. This report is to introduce a point by point portrayal of the course the executives framework. It will make sense of the reason and highlights of the framework, the points of interaction of the framework will do, the imperatives under which it should work and how the framework will respond to outer improvements. This report is expected for the two partners and designers of the framework.

Requirements Specification:

Requirement Specification provides a high secure storage to the web server efficiently. Software requirements deal with software and hardware resources that need to be installed on a serve which provides optimal functioning for the application. These software and hardware requirements need to be installed before the packages are installed. These are the most common set of requirements defined by any operation system. These software and hardware requirements provide a compatible support to the operation system in developing an application.

3.2 OPERATING ENVIRONMENT

Software Requirements

Operating System	:	Windows 8/10 and Linux/Mac
Coding Language	:	Python
Development Kit	:	IDE (Jupyter/Google Collab)

Hardware Requirements

Processor	:	Intel i3/ M1 chip
Clock Speed	:	1.7 GHz
Hard Disk	:	250 GB
RAM	:	4 GB

3.3 FUNCTIONAL REQUIREMENTS

1. Image Pre-processing
2. Segmentation
3. Feature Extraction
4. Classification
5. Train and Test Data Analysis
6. Result Analysis

3.4 NON-FUNCTIONAL REQUIREMENTS

i. Security

a. We are providing security to our application means that there should be no hacking of information..

ii. Usability

a. We should get response within seconds.
b. The application must have a simple, user friendly interface to save time and confusion.

iii. Reliability

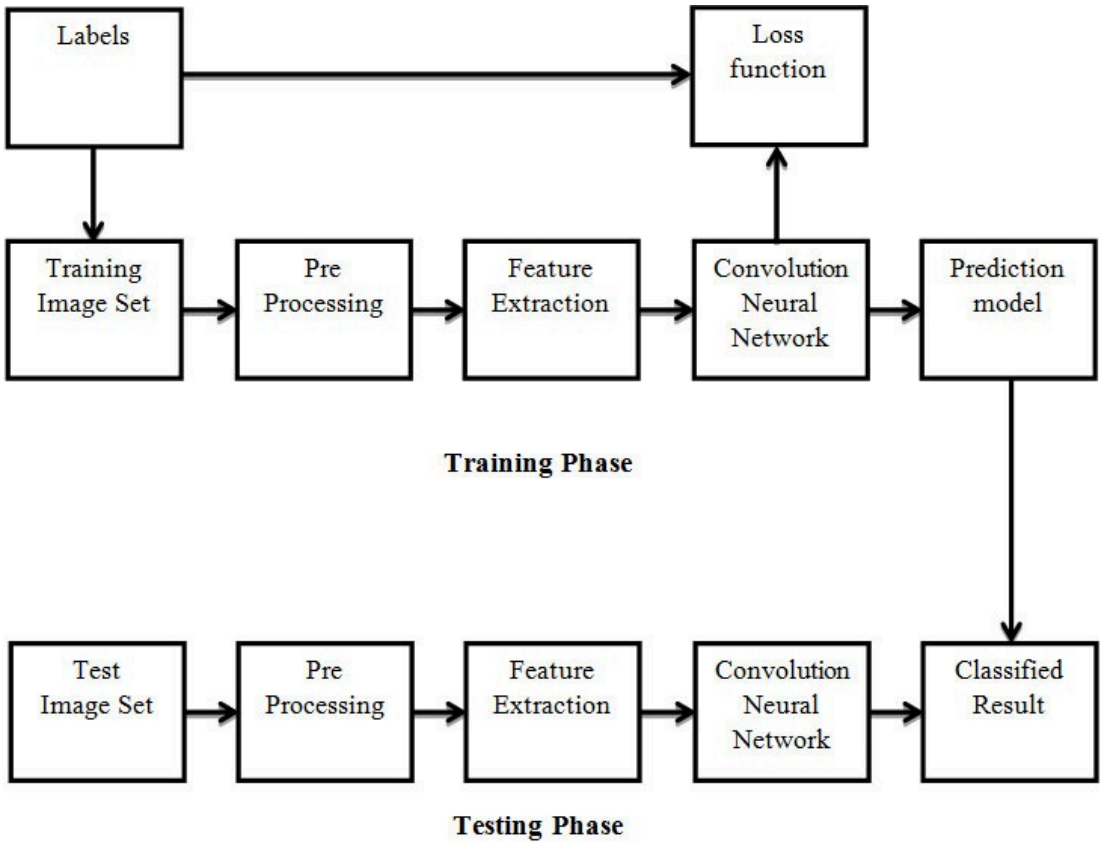
The application is reliable because of the qualities that are inherited by the reliable application standards.

iv. Performance

The application is high performing as the user's interaction with the online pharmacy management system is responded within seconds which describe the performance of the application.

4. DESIGN DIAGRAMS

4.1 SYSTEM ARCHITECTURE



As mentioned in literature review all the steps are perfectly designed to follow standard architecture depicted by the picture above to extract the results in most efficient way.

The list of UML diagrams is as follows:

- ➢ Use case diagram
- ➢ Class diagram
- ➢ Sequence diagram
- ➢ Activity diagram

4.2 Use Case Diagram

A use case describes sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse. An actor is a person, organisation or extended system that plays a role in one or more interactions with the system.

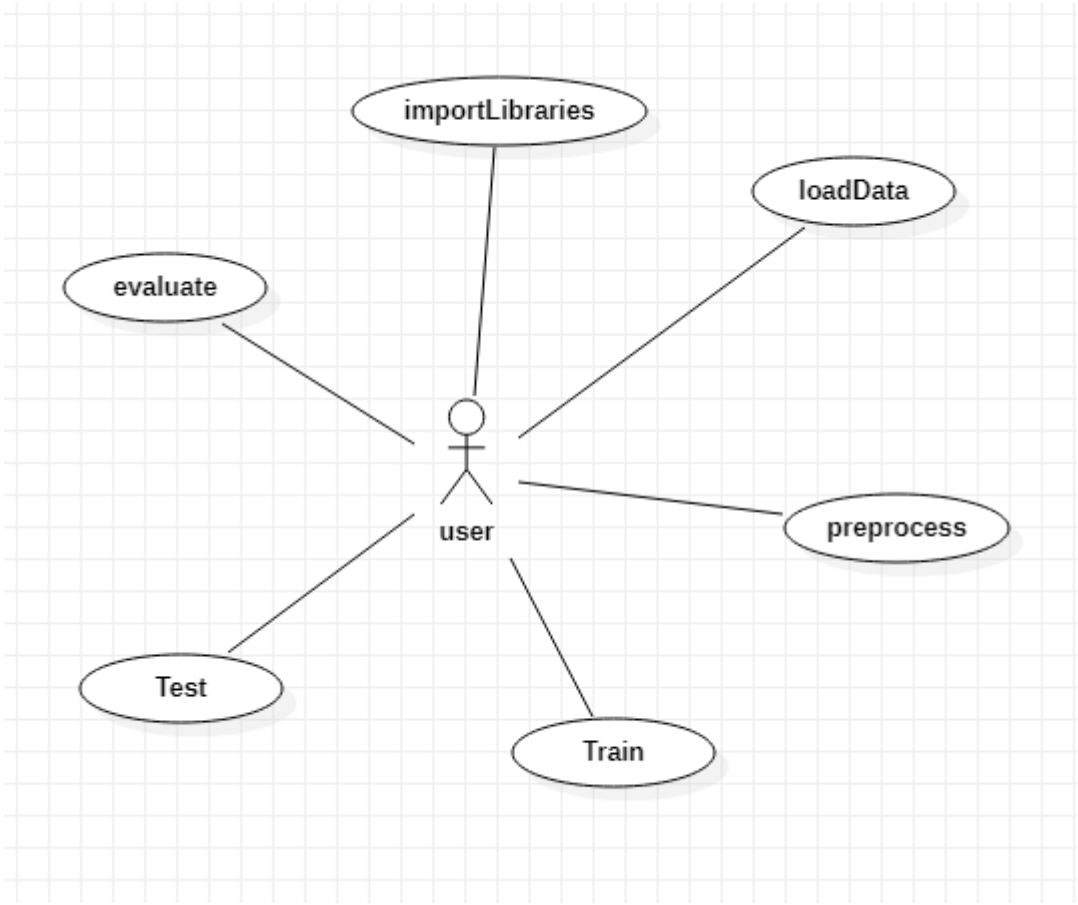


Figure 4.2 Use Case Diagram

4.3 Class Diagram

Classes are the most important building blocks of any object – oriented language. A class is a set of objects that share the same attributes, operations and relationships.

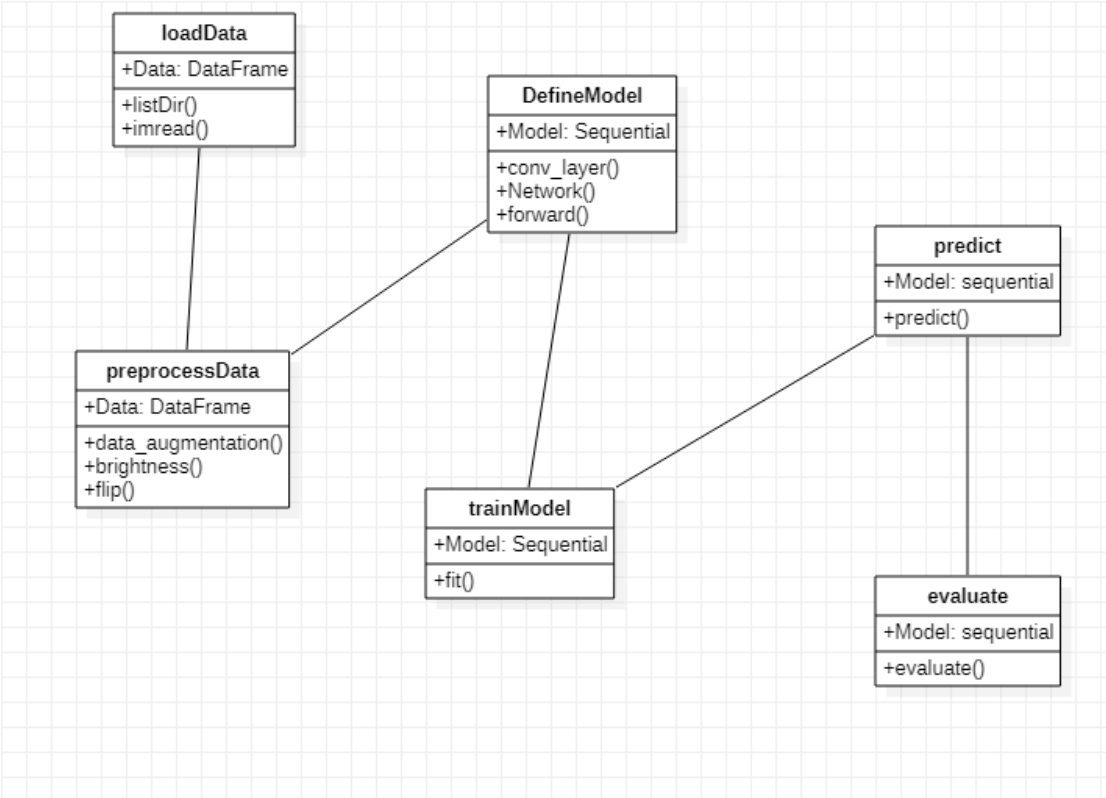


Figure 4.3 Class Diagram

4.4 Sequence Diagram

Sequence diagram model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes.

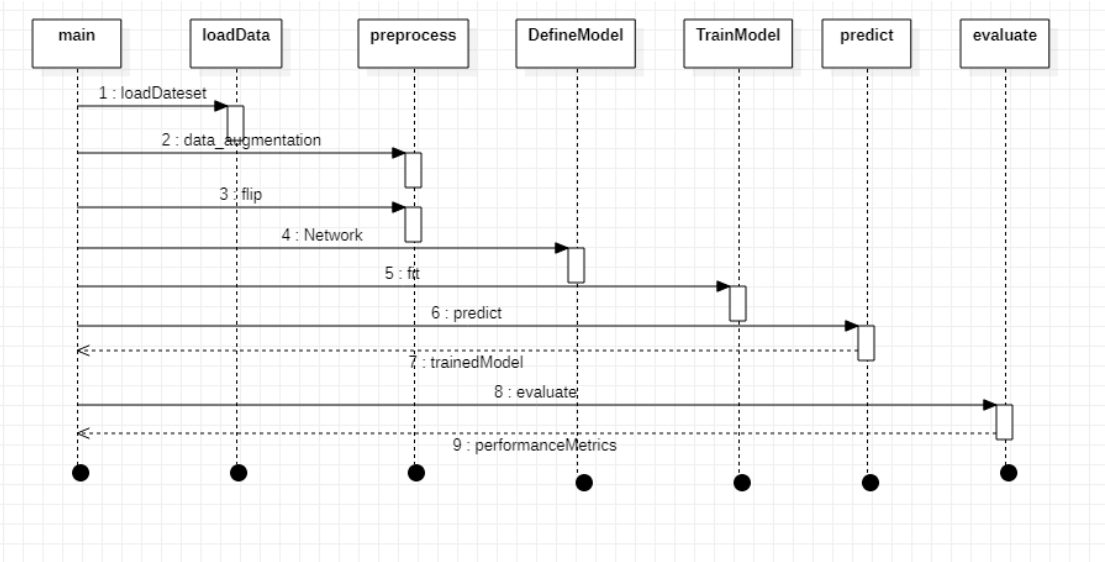


Figure 4.4 Sequence Diagram

4.5 Activity Diagram

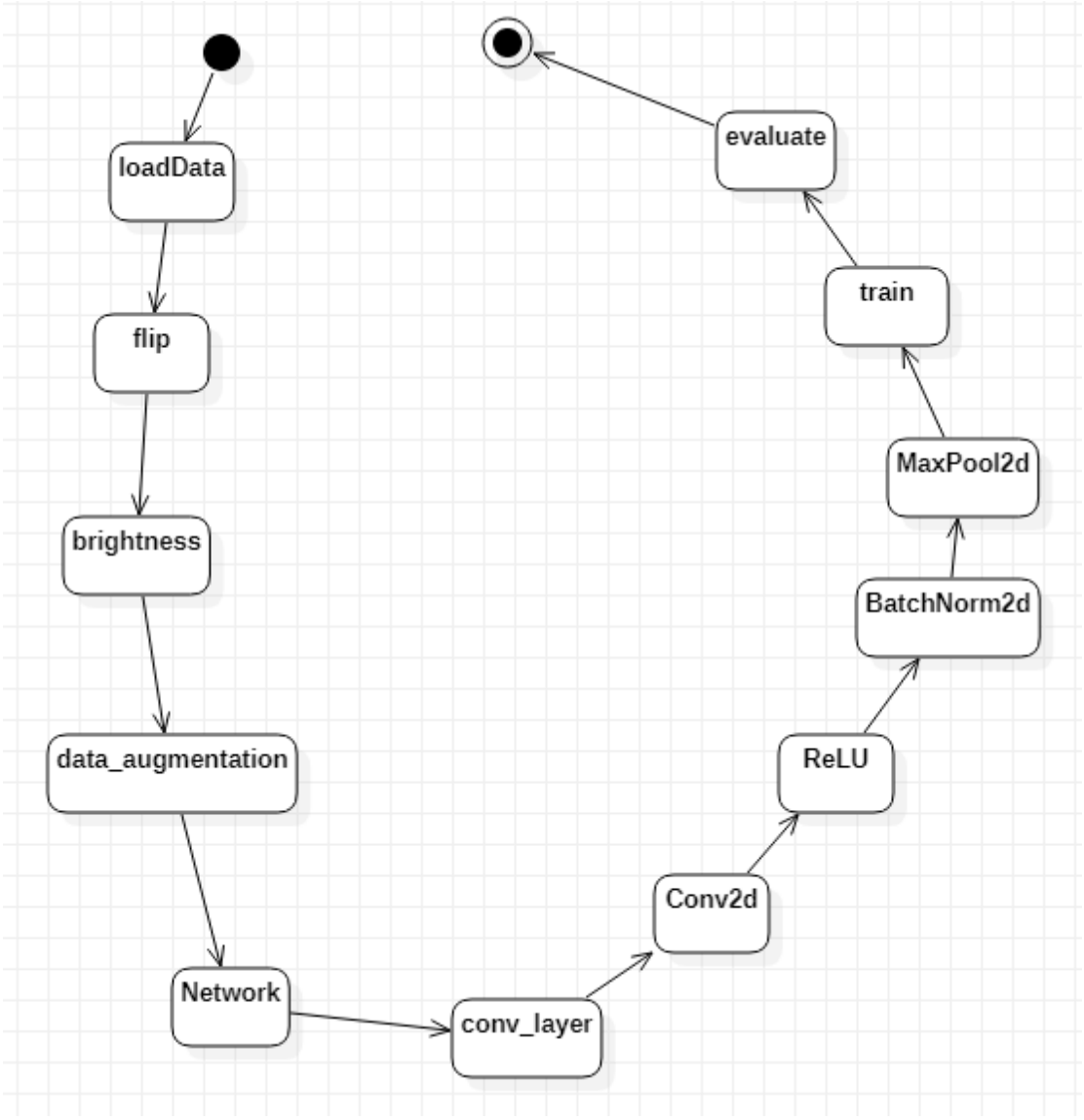


Figure 4.6 Activity diagram

5. SYSTEM IMPLEMENTATION

5.1 SAMPLE CODE

```
import numpy as np
import os
import pathlib
import skimage.io as io
import skimage.transform as tf
import skimage.exposure as ex
import skimage.color as color
import matplotlib.pyplot as plt
import torch
import torch.optim as optim
import random
import time
import copy

from google.colab import drive
drive.mount('/content/drive')

def flip(images, labels, axis):
    flipped_images = np.flip(images, axis)
    flipped_labels = labels
    return flipped_images,flipped_labels

def brightness(images, labels, gamma):
    brightness_images =  np.array([ex.adjust_gamma(image, gamma ,gain=1) for
image in images])
    brightness_labels = labels
```

```
    return brightness_images, brightness_labels


def data_augmentation(images, labels):
    # Data augmentation (flip_horizontal)
    flipped_y_images,flipped_y_labels = flip(images, labels, axis=2)

    # Concatenate arrays
    images = np.concatenate([images, flipped_y_images])
    labels = np.concatenate([labels, flipped_y_labels])

    darken_images, darken_labels = brightness(images, labels, gamma=1.5)
    brighten_images, brighten_labels = brightness(images, labels, gamma=0.5)

    # Concatenate arrays
    images = np.concatenate([images, darken_images, brighten_images])
    labels = np.concatenate([labels, darken_labels, brighten_labels])

    return images, labels


def load_data():
    path = pathlib.Path.cwd().parent / "content/drive/MyDrive/brain_tum" /"data"
    images = []
    labels = []


    for directory in os.listdir(path):
        data_path = path / directory
        for subdirectory in os.listdir(data_path):
            subdata_path = data_path / subdirectory
```

```
            for im in os.listdir(subdata_path):
                image = io.imread(f"{subdata_path}/{im}")
                image = color.rgb2gray(image)
                image = tf.resize(image, (64, 64))
                images.append(image)
                labels.append(subdirectory)

    images = np.array(images)
    labels = np.array(labels)

    images, labels = data_augmentation(images, labels)

    return images, labels

from IPython.display import Image
def predict(model,image):
    arr=image.split('/')
    img=Image(image)
    return arr[7].replace('_',' '),img

from sklearn.model_selection import train_test_split

class Data():
    def __init__(self, loader, classes):
        super(Data, self).__init__()
        self.images, self.labels = loader
        self.classes = classes
        self.shapes = (self.images.shape, self.labels.shape)
        print("Data loaded successfully :)")

    def stat(self):
```

```
    keys = self.classes.keys()
    stat = {}
    for key in keys:
        n = 0
        for label in self.labels:
            if label == key:
                n += 1
            else:
                pass
        stat[key] = n
    plt.figure(figsize=(4, 4))
    plt.pie(stat.values(), labels=stat.keys(), normalize=True)
    plt.show()

def shape(self):
    print(f"Images shape: {self.shapes[0]}",
          f"Labels shape: {self.shapes[1]}\n")

def encode(self):
    total = len(self.classes)
    self.labels = np.array([self.classes[item] for item in self.labels])
    self.labels = np.eye(total)[self.labels]

def decode(self, item):
    keys = list(self.classes.keys())
    index = np.argmax(item)
    label = keys[index]
    return label

def show(self):
    f, axis = plt.subplots(nrows=2, ncols=2, constrained_layout=True)
```

```
    for i, ax in enumerate(axis.flat):
        rand = random.randint(0, self.shapes[0][0] - 1)
        if len(self.shapes[0]) == 4:
            ax.imshow(self.images[rand])
        elif len(self.shapes[0]) == 3:
            ax.imshow(self.images[rand], cmap="gray")

        title = f"target: {self.labels[rand]}"
        ax.set_title(title)
    plt.show()

def dataset(self, split_size, shuffle, random_state, images_format,
            labels_format, permute, one_hot, device):

    if len(self.shapes[0])==3:
        self.images = np.expand_dims(self.images, axis=3)

    elif len(self.shapes[0])==4:
        pass

    if one_hot:
        self.encode()
    else:
        pass

    x_train, x_val, y_train, y_val = train_test_split(self.images,
                                                      self.labels,
                                                      test_size=split_size,
                                                      shuffle=shuffle,
                                                      random_state=random_state)
```

```
        x_test, x_val, y_test, y_val = train_test_split(x_val, y_val,
                                                        test_size=0.5,
                                                        shuffle=shuffle,
                                                        random_state=random_state)

        # Free memory
        del self.images, self.labels

        # Convert Numpy arrays to Torch tensors
        self.train_inputs = torch.from_numpy(x_train).to(images_format).to(device)
        self.train_outputs = torch.from_numpy(y_train).to(labels_format).to(device)
        del x_train, y_train

        self.val_inputs = torch.from_numpy(x_val).to(images_format).to(device)
        self.val_outputs = torch.from_numpy(y_val).to(labels_format).to(device)
        del x_val, y_val

        self.test_inputs = torch.from_numpy(x_test).to(images_format).to(device)
        self.test_outputs = torch.from_numpy(y_test).to(labels_format).to(device)
        del x_test, y_test

        if permute:
            self.train_inputs = self.train_inputs.permute(0, 3, 1, 2)
            self.val_inputs = self.val_inputs.permute(0, 3, 1, 2)
            self.test_inputs = self.test_inputs.permute(0, 3, 1, 2)

        # Verify datasets shapes
        print(f"Train tensor shape: {self.train_inputs.shape}, {self.train_outputs.shape}")
        print(f"Test tensor shape: {self.test_inputs.shape}, {self.test_outputs.shape}")
        print(f"Validation tensor shape: {self.val_inputs.shape},
{self.val_outputs.shape}")
```

```
        print("\nDataset generated successfully")

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
classes = {'glioma_tumor': 0, 'meningioma_tumor': 1, 'no_tumor': 2, 'pituitary_tumor':
3}

data = Data(loader=load_data(), classes=classes)
data.shape()
data.show()

data.dataset(split_size=0.05, shuffle=True, random_state=42,
            images_format=torch.float32, labels_format=torch.float32,
            permute=True,one_hot=True ,device=device)

# Define CNN
class conv_layer(torch.nn.Module):

    def __init__(self, in_features, out_features):
        super(conv_layer, self).__init__()
        self.conv = torch.nn.Conv2d(in_features, out_features, kernel_size=3, stride=1,
padding=1)
        self.relu = torch.nn.ReLU()
        self.norm = torch.nn.BatchNorm2d(out_features)
        self.pool = torch.nn.MaxPool2d(kernel_size=2, stride=2)



    def forward(self, X: torch.Tensor) -> torch.Tensor:
        X = self.conv(X)
        X = self.relu(X)
        X = self.norm(X)
```

```
        X = self.pool(X)
        return X


class Network(torch.nn.Module):

    def __init__(self):
        super(Network, self).__init__()
        self.input_norm = torch.nn.BatchNorm2d(1, affine=False)
        self.layer1 = conv_layer(in_features=1, out_features=8)
        self.layer2 = conv_layer(in_features=8, out_features=16)
        self.layer3 = conv_layer(in_features=16, out_features=32)
        self.layer4 = conv_layer(in_features=32, out_features=64)
        self.layer5 = conv_layer(in_features=64, out_features=128)
        self.layer6 = conv_layer(in_features=128, out_features=256)



        self.net = torch.nn.Sequential(self.layer1, self.layer2, self.layer3,
                                       self.layer4, self.layer5, self.layer6)



        self.fc1 = torch.nn.Linear(in_features=256, out_features=128)
        self.bn1 = torch.nn.BatchNorm1d(128)

        self.fc2 = torch.nn.Linear(in_features=128, out_features=32)
        self.bn2 = torch.nn.BatchNorm1d(32)

        self.fc3 = torch.nn.Linear(in_features=32, out_features=8)
        self.bn3 = torch.nn.BatchNorm1d(8)
```

```
        self.fc4 = torch.nn.Linear(in_features=8, out_features=4)


        self.lin = torch.nn.Sequential(self.fc1, self.bn1, self.fc2, self.bn2,
                                       self.fc3, self.bn3, self.fc4)


    def forward(self, X: torch.Tensor) -> torch.Tensor:
        X = self.input_norm(X)
        X = self.net(X)
        X = X.reshape(X.size(0), -1)
        X = self.lin(X)
        X = torch.nn.functional.elu(X, alpha=1.0, inplace=False)
        return X

def rmse(target, pred):
    MSE = torch.nn.functional.mse_loss(target, pred, reduction="sum")
    return torch.sqrt(MSE)

class Model():
    def __init__(self, network, optimizer, device):
        super(Model, self).__init__()
        self.net = network.to(device)
        self.optim = optimizer
        self.device = device
        print("Model initialized succssefully :)\n")

    def train(self, train_data, val_data, epochs, patience, batch_size, learning_rate):
        if self.optim == "adam":
            self.optim = torch.optim.Adam(self.net.parameters(), lr=learning_rate)
        best_loss = np.inf
```

```
self.patience = patience
self.train_losses = []
self.val_losses = []
self.achieved_epochs = []
train_inputs, train_outputs = train_data
val_inputs, val_outputs = val_data
total_train = train_inputs.size()[0]
total_val = val_inputs.size()[0]
print("Train loop:\n")

t0 = time.time()
for epoch in range(epochs):
    self.net.train()
    train_loss = 0
    val_loss = 0
    self.achieved_epochs.append(epoch)
    train_permutation = torch.randperm(total_train)
    val_permutation = torch.randperm(total_val)

    for i in range(0, total_train, batch_size):
        self.optim.zero_grad()
        indices = train_permutation[i:i+batch_size]
        batch_x, batch_y = train_inputs[indices], train_outputs[indices]
        outputs = self.net(batch_x)
        loss = rmse(outputs, batch_y)
        loss.backward()
        self.optim.step()
        train_loss += loss
    train_loss = train_loss.cpu().detach() / total_train
    self.train_losses.append(train_loss)
```

```
        for j in range(0, total_val, batch_size):
            self.net.eval()
            indices = val_permutation[j:j+batch_size]
            batch_x, batch_y = val_inputs[indices], val_outputs[indices]
            outputs = self.net(batch_x)
            loss = rmse(outputs, batch_y)
            val_loss += loss
        val_loss = val_loss.cpu().detach() / total_val
        self.val_losses.append(val_loss)

        if val_loss < best_loss:
            best_loss = val_loss
            cost_patience = patience
            self.state_dict = copy.deepcopy(self.net.state_dict())
            print(f"\tEpoch: {epoch+1}/{epochs}, ",
                  f"Train Loss: {train_loss:.3g}, ",
                  f"Val Loss: {val_loss:.3g}")

        else:
            cost_patience -= 1
            if cost_patience < 0:
                print(f"\nEarly stopping after {patience} epochs of no improvements")
                break

            else:
                print(f"\tEpoch: {epoch+1}/{epochs}, ",
                      f"Train Loss: {train_loss:.3g}, ",
                      f"Val Loss: {val_loss:.3g} - No improvement",
                      f"-> Remaining patience: {cost_patience}")

    tf = time.time()
```

```
    print(f"\nTrain finished successfully in {tf-t0:.3g}s")

def evaluate(self, test_data):
    test_inputs, test_outputs = test_data
    self.net.load_state_dict(self.state_dict)
    predictions = self.net(test_inputs).cpu().detach().numpy()
    correct = 0
    wrong = 0
    for i,(j,k) in enumerate(zip(predictions, test_outputs.cpu().detach())):
        if np.argmax(j) == np.argmax(k):
            correct +=1
        else:
            wrong += 1

    score = 100 * correct / test_outputs.shape[0]
    print(f'\nTest accuracy:{score:.3g}%')
    print(f'Correct predictions: {correct}, Wrong predictions: {wrong}')

def save(self, path, checkpoint_name):
    torch.save(self.state_dict, f"{path}/{checkpoint_name}.pth")
    print("\nCheckpoint saved successfully :)")

def plot(self):
    f, ax = plt.subplots()
    ax.plot(self.achieved_epochs, self.train_losses, label='train')
    ax.plot(self.achieved_epochs, self.val_losses, label='validation')
    ax.set_title('model loss')
    ax.set_ylabel('loss')
    ax.set_xlabel('epoch')
    no_improvement_line = self.achieved_epochs[-1] - self.patience
    ax.axvline(x=no_improvement_line, color='r')
```

```
        ax.legend(loc='upper center', frameon=False)
        plt.show()

net = Network()
optimizer = optim.Adam(net.parameters(), lr=1.0E-3)

BrainTumorClassifier = Model(net, optimizer, device)
BrainTumorClassifier.train(train_data=(data.train_inputs, data.train_outputs),
                           val_data=(data.val_inputs, data.val_outputs),
                           epochs=20, patience=5, batch_size=100,learning_rate=1.0E-3)

BrainTumorClassifier.evaluate(test_data=(data.test_inputs, data.test_outputs))

BrainTumorClassifier.plot()
BrainTumorClassifier.save(path="./", checkpoint_name="module")



# import tkinter module
import tkinter
from tkinter import *
from tkinter import font
from tkinter.ttk import *
from tkinter.filedialog import askopenfilename
import pandas as pd
import numpy as np
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from typing import Counter
from textblob import TextBlob
from googletrans import Translator
import warnings
```

```
from PIL import Image, ImageOps
from tensorflow.keras.models import load_model
warnings.filterwarnings("ignore")
# creating main tkinter window/toplevel
master = Tk()
master.geometry("500x200")
# master.state("zoomed")
master.title("Brain tumor classification using MRI images")
text = tkinter.Label( text="Brain tumor classification using MRI images",
font=("Helvetica", 12), height=2, anchor='n', fg='#f00')
def askopenfile():
    global filename
    global df
    global name
    filename=askopenfilename( filetypes =(("JPG Files","*.jpg"),))
    result['text']="File uploaded"
b=Button(text="Upload Image", command=askopenfile)
result = tkinter.Label(text="you will see result here!",font=('Courier', 12), height=40,
anchor='nw')
text.grid(row=0, columnspan=4)

def Close():
    master.destroy()
def predict():
    labels=['No tumor','meningioma tumor','glioma tumor','pituitary tumor']
    model = load_model('model.h5')
    data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)
    image = Image.open(filename)
    size = (224, 224)
    image = ImageOps.fit(image, size, Image.ANTIALIAS)
```

```
    image_array = np.asarray(image)
    normalized_image_array = (image_array.astype(np.float32) / 127.0) - 1

    data[0] = normalized_image_array

    prediction = model.predict(data)



    result['text']=labels[np.argmax(prediction)]
b1=Button(text="predict", command=predict)
b2=Button(text="Close application", command=Close)
b.grid(row=1,column=0)
b1.grid(row=2,column=0)
b2.grid(row=3, column=0)
result.grid(rowspan=6,columnspan=4)

# infinite loop which can be terminated
# by keyboard or mouse interrupt
mainloop()
```

6. SYSTEM TESTING

Testing involves the execution of a software component or system component to evaluate one or more properties of interest with respect to our project, Online pharmacy management system. In general, these properties indicate the extent to which the project is tested:

(i) Meets the requirements that guided its design and development
(ii) Responds correctly to all kinds of inputs.
(iii) Performs its functions within an acceptable time.
(iv) Sufficiently usable.
(v) Can be opened and run in intended environments
(vi) Achieves the general result

6.1 TEST CASES

In general, a test case is a set of test data and test programs and their expected results. A test case in software engineering normally consists of a unique identifier, requirement references from a design specification, preconditions, events, a series of steps (also known as actions) to follow, input, output and it validates one or more system requirements and generates a pass or fail.

The mechanism for determining whether a software program or system has passed or failed such a test is known as a test case. It may take many test cases to determine that a software program or system is considered sufficiently scrutinized to be released. Test cases are often referred to as test scripts, particularly when written. Written test cases are usually collected into test suites. If a requirement has sub-requirements, each sub-requirement must have at least two test cases. Written test cases should include a description of the functionality to be tested, and the preparation required to ensure that the test can be conducted.

6.1.1 Test Case 1

```
In [12]:
1  out = model.predict_segmentation(
2      inp=valid_dir+"/3.png",
3      out_fname=valid_dir+"/predict/output.png"
4  )
```

```
In [13]:
1  plt.imshow(out)
```

```
Out[13]: <matplotlib.image.AxesImage at 0x7f0afbcb7290>
```

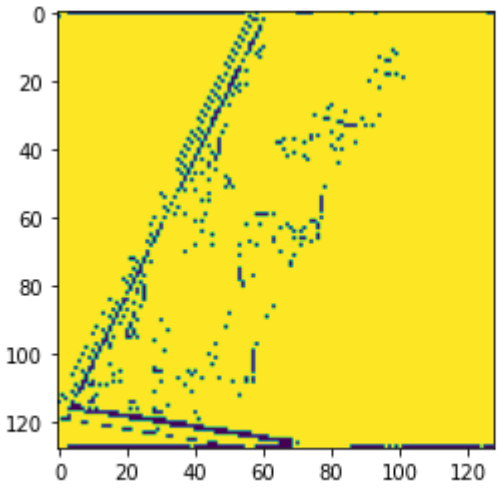


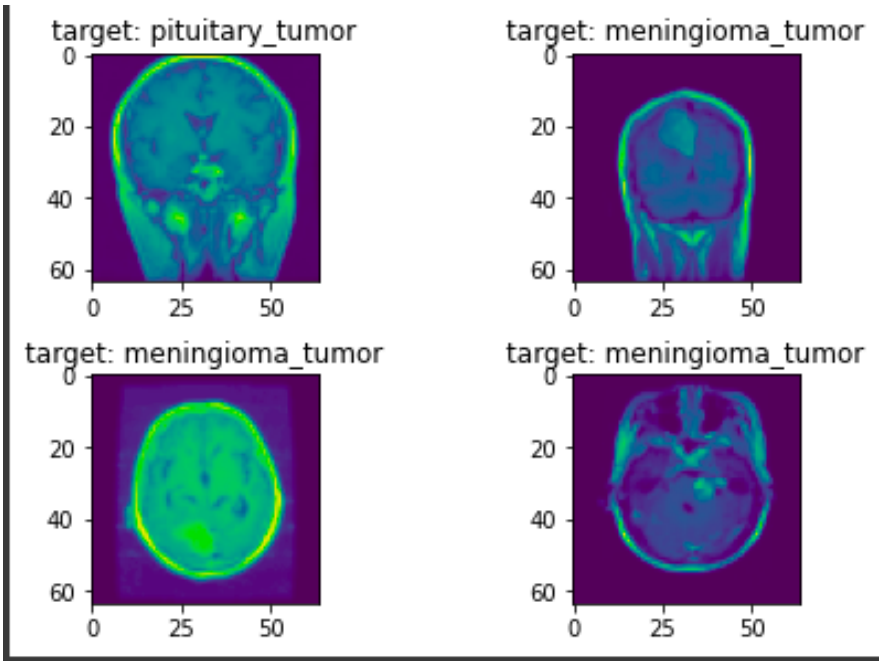
Fig 6.3 noYawn

7. SCREENSHOTS

7.1 Epoch Report

```
Model initialized succssefully :)

Train loop:

        Epoch: 1/20,  Epoch duration: 45s,  Train Loss: 0.0796,  Val Loss: 0.0703
        Epoch: 2/20,  Epoch duration: 43.6s,  Train Loss: 0.05,  Val Loss: 0.0457
        Epoch: 3/20,  Epoch duration: 44.1s,  Train Loss: 0.0359,  Val Loss: 0.0411
        Epoch: 4/20,  Epoch duration: 43.2s,  Train Loss: 0.0285,  Val Loss: 0.0372
        Epoch: 5/20,  Epoch duration: 43.1s,  Train Loss: 0.0232,  Val Loss: 0.0273
        Epoch: 6/20,  Epoch duration: 43.8s,  Train Loss: 0.0191,  Val Loss: 0.0229
        Epoch: 7/20,  Epoch duration: 43.5s,  Train Loss: 0.0169,  Val Loss: 0.0234 - No improvement -> Remaining patien
        Epoch: 8/20,  Epoch duration: 43.4s,  Train Loss: 0.0165,  Val Loss: 0.0222
        Epoch: 9/20,  Epoch duration: 43.2s,  Train Loss: 0.0143,  Val Loss: 0.0198
        Epoch: 10/20,  Epoch duration: 42.7s,  Train Loss: 0.0145,  Val Loss: 0.0182
        Epoch: 11/20,  Epoch duration: 42.8s,  Train Loss: 0.0137,  Val Loss: 0.0222 - No improvement -> Remaining patie
        Epoch: 12/20,  Epoch duration: 42.7s,  Train Loss: 0.0129,  Val Loss: 0.0212 - No improvement -> Remaining patie
        Epoch: 13/20,  Epoch duration: 42.5s,  Train Loss: 0.0122,  Val Loss: 0.0208 - No improvement -> Remaining patie
        Epoch: 14/20,  Epoch duration: 42.6s,  Train Loss: 0.0106,  Val Loss: 0.0212 - No improvement -> Remaining patie
        Epoch: 15/20,  Epoch duration: 42.7s,  Train Loss: 0.0282,  Val Loss: 0.0266 - No improvement -> Remaining patie

Early stopping after 5 epochs of no improvements

Train finished successfully :)
```

7.2 Output

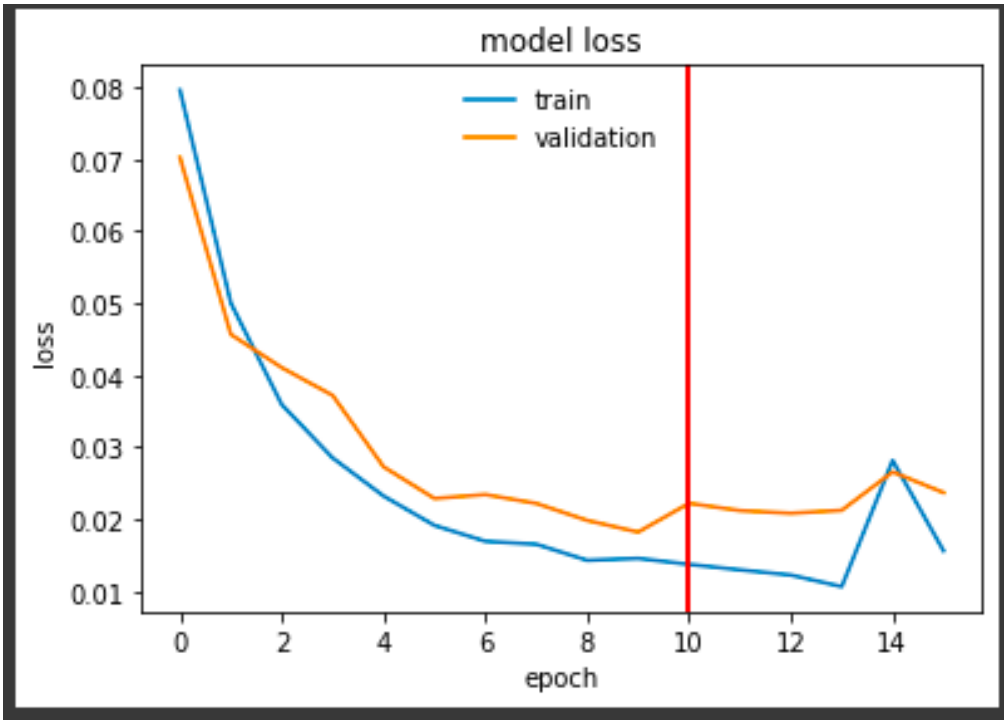


Pre-processed Image

Glioma Tumor Detected

No Tumor Detected

7.3 Training and Validation Loss Graph



7.3 Sample GUI

8. CONCLUSION

This project address problems with existing system and solves them effectively. This a fully functional model that efficiently detects brain tumours and Convolutional neural networks represent a growing field that will likely help radiologists provide more accurate care for their patients.In the end, we have achieved a fully functional model that efficiently detects brain tumours.

FUTURE ENHANCEMENTS

Updates are best to continue the legacy of any applications. For this we propose to integrate into MRI scanners with automatic contours around t area. Through the further development of segmentation techniques in brain tumours, this could be applied to other areas of radiology.Through the further development of segmentation techniques in brain tumours, this could be applied to other areas of radiology.

9. BIBLIOGRAPHY

[1] Hany Kasban, Mohsen El-bendary, Dina Salama, A comparative study of medical imaging techniques.

J. Clerk Maxwell, A Treatise on Electricity and Magnetism.

[2] M.L. Oelze, J.F. Zachary, W.D. O'Brien Jr., Differentiation of tumour types in vivo by scatterer property estimates and parametric images using ultrasound backscatter.

[3] Brain Tumour: Statistics, Cancer.Net Editorial Board.

[4] A. Hamamci, et al., Tumour-Cut: segmentation of brain Tumours on contrast enhanced MR images for radiosurgery applications, IEEE Trans. Med. Imaging 31 (3) (2012) 790–804.

[10] M. Havaei, H. Larochelle, P. Poulin, P.M. Jadoin, Within-brain classification for brain tumour segmentation.

REFERENCES

- https://github.com/SartajBhuvaji/Brain-Tumor-Classification-DataSet
- https://ieeexplore.ieee.org/document/8934561
- https://scikit-learn.org
- https://scikit-learn.org/0.21/documentation.html
- https://pytorch.org/docs/stable/index.html