

BRAIN TUMOUR DETECTION USING CNN

*A Mini Project-I Report submitted
in partial fulfillment of the requirements
for the award of the degree of*

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

By

1. K.Arthi-	19B01A0574
1. K.Lovely Srenika -	19B01A0583
2. M.Bhavana Swetha -	19B01A0591
3. P.S.S.L.Srihitha-	19B01A0599
4. N.L.S.Akhila -	19B01A05C6

Under the esteemed guidance of
Mrs.G.R.L.M.Tayaru



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN (A)
(Approved by AICTE, Accredited by NBA & NAAC, Affiliated to JNTU Kakinada)
BHIMAVARAM – 534 202
2021 – 2022

SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN (A)
(Approved by AICTE, Accredited by NBA & NAAC, Affiliated to JNTU Kakinada)
BHIMAVARAM – 534 202

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



CERTIFICATE

*This is to certify that the Mini Project-I entitled "**BRAIN TUMOR DETECTION USING CNN**", is being submitted by **K.ARTHI, K.LOVELY SRENIKA, M.BHAVANA SWETHA P.S.S.L.SRIHITHA, N.L.S.AKHILA** bearing the **Regd.No. 19B01A0574, 19B01A0583, 19B01A0591, 19B01A0599, 19B05A05C6** in partial fulfillment of the requirements for the award of the degree of "**Bachelor of Technology in Computer Science & Engineering**" is a record of bonafide work carried out by her under my guidance and supervision during the academic year **2021 – 2022** and it has been found worthy of acceptance according to the requirements of the university.*

Internal Guide

Head of the Department

ACKNOWLEDGEMENT

We sincerely express indebtedness to esteemed and revered guide **Mrs.G.R.L.M.Tayaru** of Department of Computer Science and Engineering for his valuable guidance, supervision, and encouragement throughout the work. Without his kind patronage and guidance, the synopsis would not have taken shape.

We take this opportunity to express a deep sense of gratitude to our respected chairman sir Sri **K.V.Vishnu** Raju, our principal sir **Dr.G.Srinivasa Rao**, vice-Principal sir **Dr.P.Srinivasa Raju, Dr.P.Kiran sree sir (HOD)**, Head of Department of Computer Science and Engineering, and **Mrs. T. Gayatri (Project Co-Ordinator)** and **Mrs.G.R.L.M.Tayaru (Project Guide) - PRC Members** for their encouragement and kind approval. We would like to express our sincere regards to them for advice and counseling from time to time. We owe sincere thanks to all the faculties in Department of Computer Science and Engineering for their advice and counseling time to time.

K.Arthi - 19B01A0574

K.Lovely Srenika - 19B01A0583

M.Bhavana Swetha-19B01A0591

P.S.S.L.Srihitha-19B01A0599

N.L.S.Akhila-19B01A05C6

ABSTRACT

The brain tumors, are the most common and aggressive disease, leading to a very short life expectancy in their highest grade. Thus, treatment planning is a key stage to improve the quality of life of patients. Generally, various image techniques such as Computed Tomography (CT), Magnetic Resonance Imaging (MRI) and ultrasound image are used to evaluate the tumor in a brain. Especially, in this work MRI images are used to diagnose tumor in the brain.

The aim of the project is to develop a “**Brain Tumor Detection Model**” which detects the brain tumor from the given MRI Images using **CNN**.

In this work, automatic brain tumor detection is proposed by using Convolutional Neural Networks (CNN) classification. The deeper architecture design is performed by using small kernels. The weight of the neuron is given as small.

The objective of this project is to increase the accuracy, decrease the complexity.

Dataset Specifications: The Brain Tumor dataset is taken from Kaggle. It contains 98 files which does not contain tumor and 155 files which contains tumor.

Percentage used for training is 80% and percentage used for testing is 20%.

Contents

S.No.	PageNo.
1. Introduction	6
2. System Analysis	7-8
2.1 Existing System	
2.2 Proposed System	
2.3 Feasibility Study	
3. System Requirements Specification	9-11
3.1 Software Requirements	
3.2 Hardware Requirements	
3.3 Functional Requirements	
4. System Design	12-18
4.1 Introduction	
4.2 UML Diagrams	
4.3 Database Design	
4.3.1 ER Diagrams	
5. System Implementation	19-29
5.1 Introduction	
5.2 Project Modules	
6. System Testing	30-35
6.1 Introduction	
6.2 Testing Methods	
7. Conclusion	36
8. Bibliography	37
9. Appendix	38
9.1 Introduction to Machine Learning	

1. INTRODUCTION

The brain tumors, are the most common and aggressive disease, leading to a very short life expectancy in their highest grade. Thus, treatment planning is a key stage to improve the quality of life of patients. Generally, various image techniques such as Computed Tomography (CT), Magnetic Resonance Imaging (MRI) and ultrasound image are used to evaluate the tumor in a brain. Especially, in this work MRI images are used to diagnose tumor in the brain.

The aim of the project is to develop a "Brain Tumor Detection Model" which detects the brain tumor from the given MRI Images using CNN.

In this work, automatic brain tumor detection is proposed by using Convolutional Neural Networks (CNN) classification. The deeper architecture design is performed by using small kernels. The weight of the neuron is given as small.

The objective of this project is to increase the accuracy, decrease the complexity.

Dataset Specifications: The Brain Tumor dataset is taken from Kaggle. It contains 98 files which does not contain tumor and 155 files which contains tumor.

Percentage used for training is 80% and percentage used for testing is 20%.

2. SYSTEM ANALYSIS

2.1. EXISTING SYSTEM

- In,1 the Fuzzy C-Means (FCM) segmentation is applied to separate the tumor and non-tumor region of brain. An accuracy rate of 96.97% in the analysis of DNN based brain tumor classification.
- The Fuzzy Inference System (FIS) is a one special technique, which is mainly used for brain segmentation. Supervised classification is used to create a membership function of fuzzy controller. The performance is high and accuracy is low.
- Finally, the fuzzy with K Nearest Neighbor (KNN) classification is applied to find the abnormality of brain MRI image. The complexity is high. But the accuracy is low. In this work, a novel automatic brain tumor classification is performed by convolutions neural network.

2.2. PROPOSED SYSTEM

- The main goal of this research work is to design efficient automatic brain tumor detection with high accuracy, performance and low complexity.
- Python language is used for implementation.
- CNN is one of the deep learning methods, which contains sequence of feed forward layers.
- Following are the benefits of using CNN:
 - ✓ To improve the accuracy and to reduce the computation time, a convolution neural network-based classification (**CNN**) is introduced in the proposed scheme.
 - ✓ Image net database is used for classification. It is one of the pre-trained models. So, the training is performed for only final layer. Also, raw pixel value with depth, width and height feature value are extracted from CNN.
 - ✓ The training accuracy, validation accuracy and validation loss are calculated. The training accuracy is obtained by using CNN model. Similarly, the validation accuracy will be high and validation loss will be very low.
- Also, the detection results are given as whether tumor is present or not in the given MRI image of the brain

2.3. FEASIBILITY STUDY

A feasibility study is an analysis that consider all of a project's relevant factors including economic, technical, legal, and scheduling considerations to ascertain the likelihood of completing the project successfully.

However, a feasibility study isn't only used for projects looking to measure and forecast financial gains. In other words, feasible can mean something different, depending on the industry and the project's goal.

Technical Feasibility

Technical feasibility deals with the existing technology, software and hardware requirements for the proposed system. The proposed system "**BRAIN TUMOR DETECTION USING CNN**" is planned to run on Python.

3. SYSTEM REQUIREMENTS SPECIFICATION

3.1. SOFTWARE REQUIREMENTS

- Operating System: any Windows OS
- Libraries: Keras, Tensor Flow, Numpy, Scikit-Learn, Matplotlib, OpenCV
- Editor: Collab Notebook
- Technologies: Python

3.2. HARDWARE REQUIREMENTS

- Processor: i3
- RAM: 4 GB
- Hard disk: 512GB

3.3. FUNCTIONAL 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. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

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

- Pandas:

Pandas is the most popular python library that is used for data analysis. It provides highly optimized performance with back-end source code is purely written in C or Python. We can analyze data in pandas with

1. Series
2. Data frames

- Anaconda:

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution includes data-science packages suitable for Windows, Linux, and macOS. Anaconda distribution comes with 1,500 packages selected from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command-line interface (CLI).

- Jupyter Notebook:

Anaconda distribution comes with 1,500 packages selected from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI). A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text, mathematics, plots and rich media, usually ending with the ". ipynb" extension.

- Tensor Flow:

Tensor flow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

- 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.

3.3.1 HARDWARE CONFIGURATION

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.

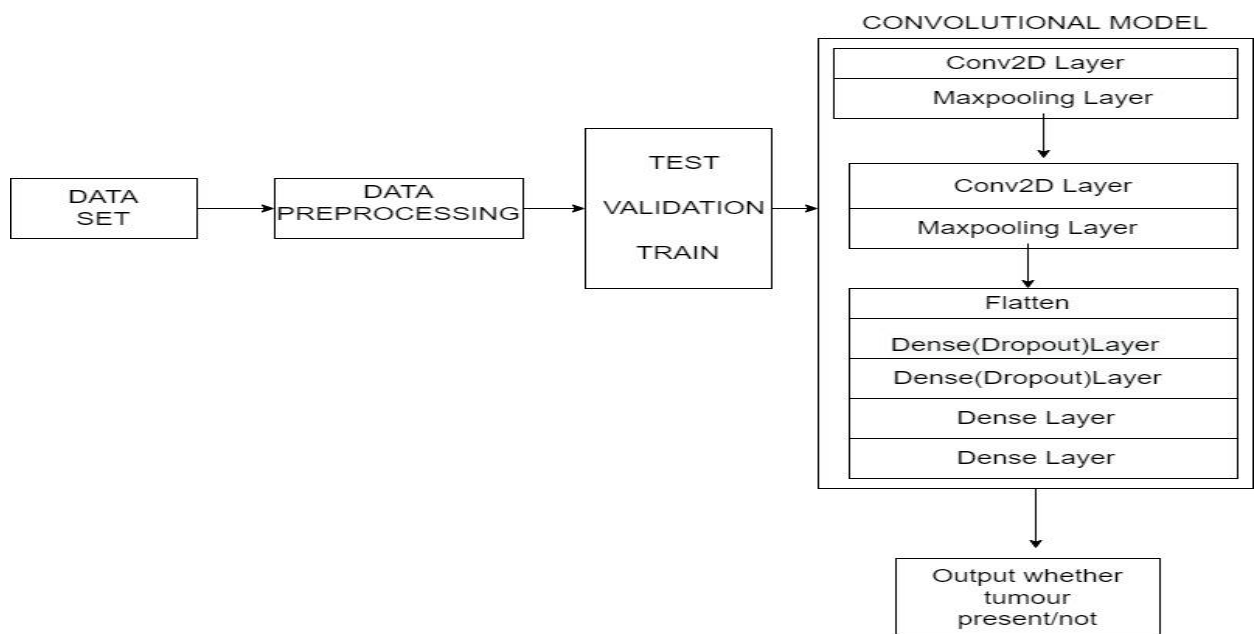
Display: Dual XGA (1024 x 768) or higher resolution monitors

Operating system: Windows

4. SYSTEM DESIGN

4.1. INTRODUCTION

Design is the first step in the development phase of an engineering product or system. Design is the place where quality is considered in the software development. Design is the only way that we can accurately translate user requirements into finished software product or system. Software design serves as the foundation for all the software engineers and software maintenance that steps follow.



4.2. UML DIAGRAMS (Data flow diagrams)

INTRODUCTION

A model is an abstract representation of system, constructed to understand the system priority to building or modifying it. A model is a simplified representation of reality and it provides a means for conceptualization and communication of ideas in a precise and ambiguous form.

UML (Unified Modeling Language) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. It is a method for describing the system architecture in detail using the blueprint. We use UML diagrams to portray the behavior and structure of a system. This is the step while developing any product after analysis. The representation of the entities that are to be used in the product being developed need to be designed.

There are various kinds of methods in software design:

- Use case Diagram
- Class Diagram
- Sequence Diagram
- Activity Diagram
- State Chart Diagram

Use Case Diagram:

Use Case Diagrams are used to depict the functionality of a system or a part of a system. They are widely used to illustrate the functional requirements of the system and its interaction with external agents (actors).

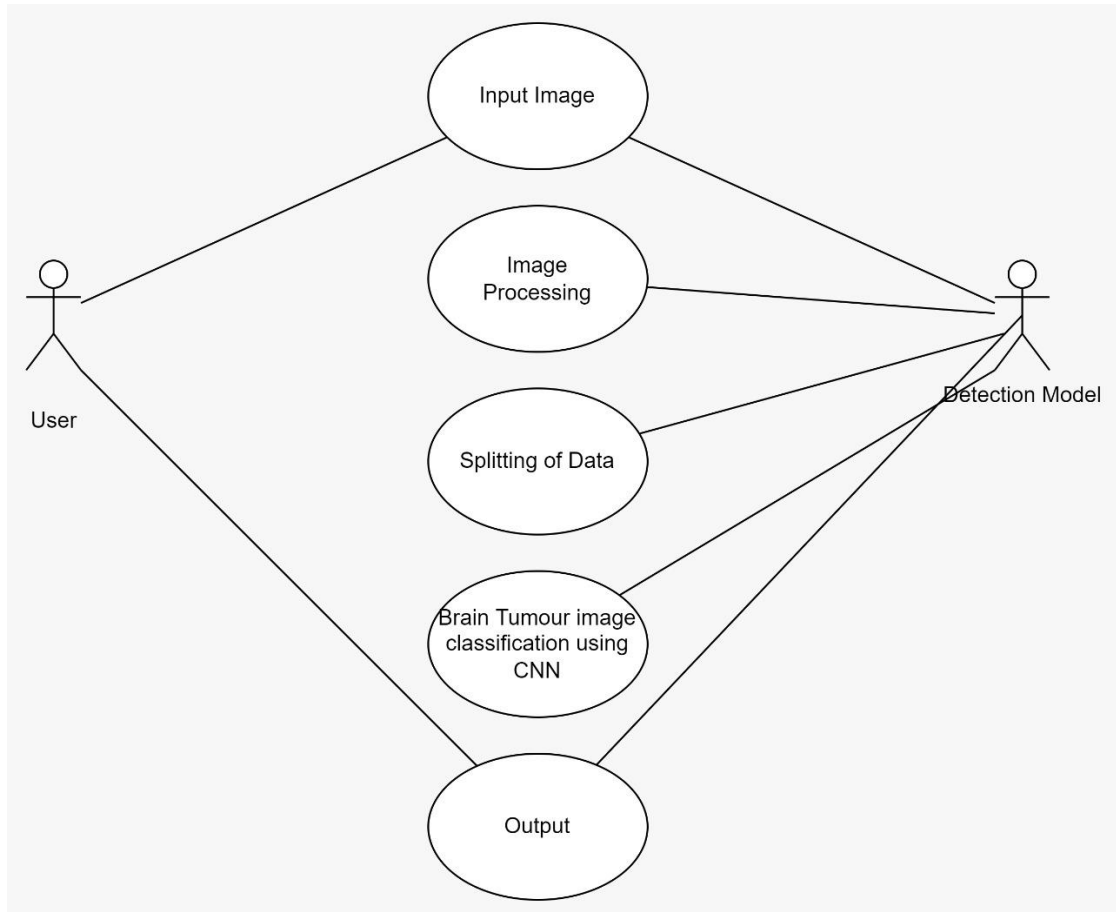
In brief, the purposes of use case diagrams can be said to be as follows

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.

Use case diagrams commonly contains

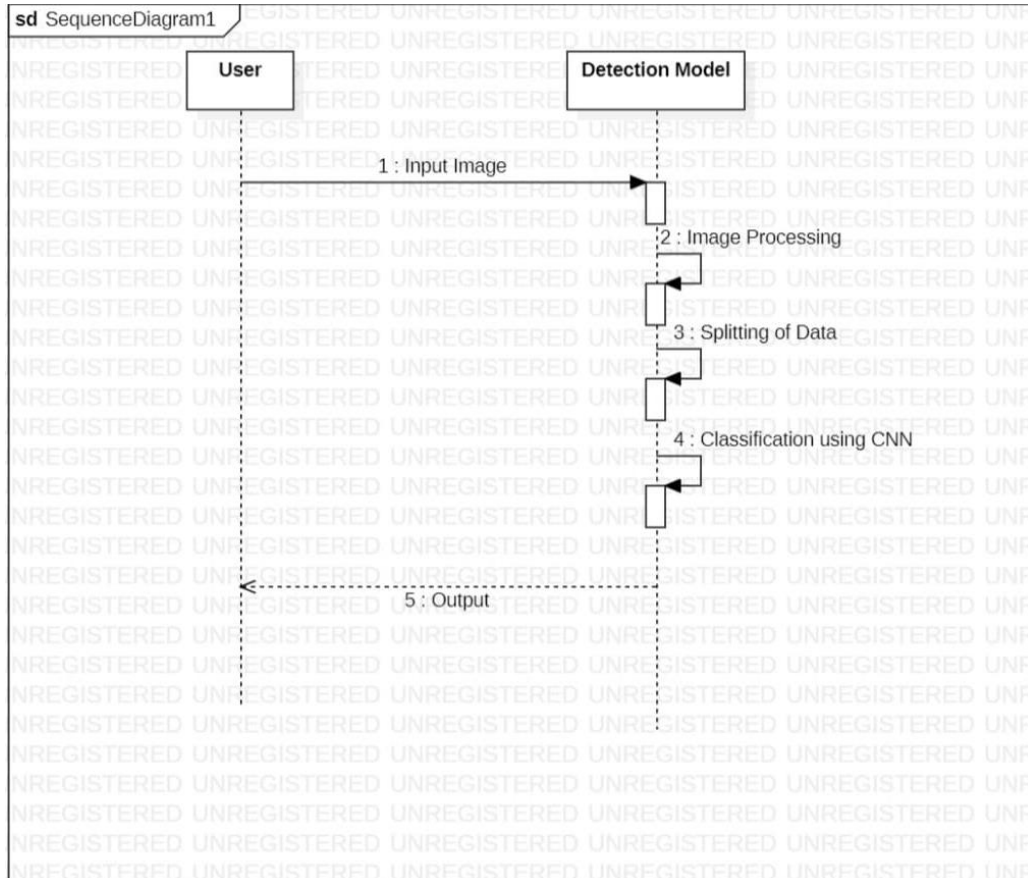
Brain Tumor detection using CNN

- Use cases
- Actors
- Dependency, generalization and association relationships.



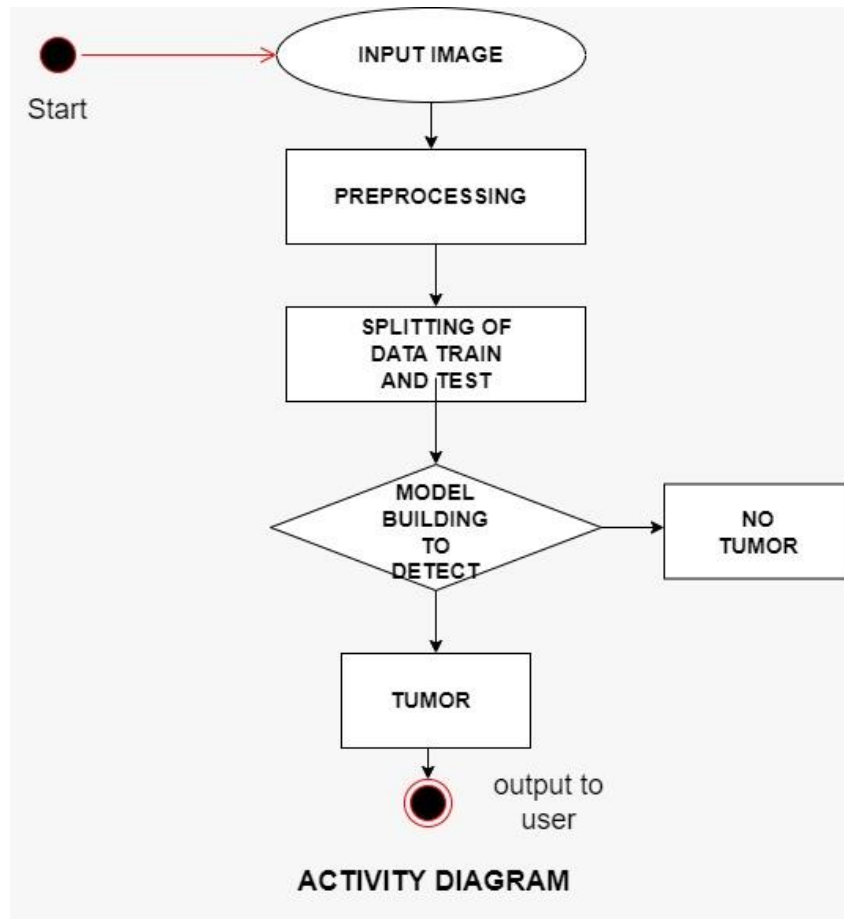
Sequence Diagram:

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. Sequence diagrams are used to formalize the behavior of the system and to visualize the communication among objects.



Activity Diagram:

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. This flow can be sequential, branched, or concurrent. The basic purposes of activity diagram is similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

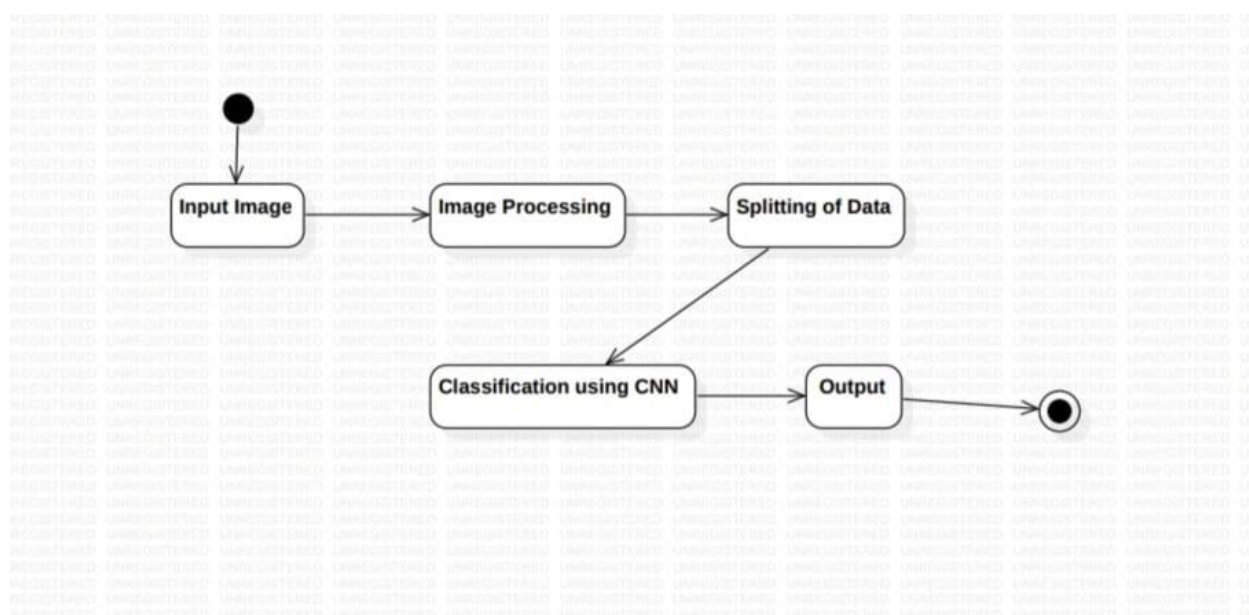


- Describe the sequence from one activity to another.
- Draw the activity flow of a system.
- Describe the parallel, branched and concurrent flow of the system.

State Chart Diagram:

State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State chart diagram is to model lifetime of an object from creation to termination. State chart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

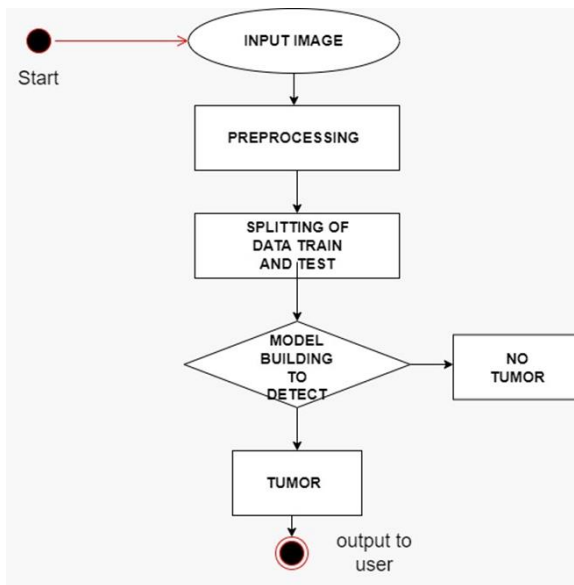
Brain Tumor detection using CNN



4.3 DATABASE DESIGN

4.3.1 ER-diagram:

An entity-relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database. ER diagrams often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes.



5. SYSTEM IMPLEMENTATION

5.1. INTRODUCTION

The purpose of system implementation can be summarized as follow making the new system available to the prepared set of users (the deployment), and positioning on-going support and maintenance of the system within the performing organization (the transaction). At a finer necessary to educate the consumer on the use of system, placing the newly developed system into production, confirming that business functions that interact with the system and functioning properly. Transitioning the system support responsibilities involve changing from a system development to the system and maintenance mode of operation, with ownership of the new system moving from the project team to the performing organization.

A key difference between system implementation and all other phases of lifecycle is that all project activities up to this point have been performed in safe, protected and check your environments. It is through the careful planning, execution and management of system implementation activities that the project team can minimize the likelihood of these occurrences and determine appropriate contingency plans in the event of the problem.

Our project explores Implementation of our system consists of a detection of tumor in brain.

```
[1] !pip install -q kaggle
```

```
[2] from google.colab import files  
files.upload()
```

```
[3] ! mkdir ~/.kaggle
```

```
[4] ! cp kaggle.json ~/.kaggle/
```

```
[5] ! chmod 600 ~/.kaggle/kaggle.json
```

```
[6] ! kaggle datasets download -d navoneel/brain-mri-images-for-brain-tumor-detection
```

```
[7] !unzip brain.zip
```

```
[8] import os  
import keras
```

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout,
BatchNormalization
from PIL import Image
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('dark_background')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder

[9] encoder = OneHotEncoder()
encoder.fit([[0], [1]])

[10] data = []
paths = []
result = []

for r, d, f in os.walk(r'/content/yes'):
    for file in f:
        if '.jpg' in file:
            paths.append(os.path.join(r, file))

for path in paths:
    img = Image.open(path)
    img = img.resize((128,128))
    img = np.array(img)
    if(img.shape == (128,128,3)):
        data.append(np.array(img))
        result.append(encoder.transform([[0]]).toarray())

[11] paths = []
for r, d, f in os.walk(r"/content/no"):
    for file in f:
        if '.jpg' in file:
            paths.append(os.path.join(r, file))

for path in paths:
    img = Image.open(path)
    img = img.resize((128,128))
    img = np.array(img)
    if(img.shape == (128,128,3)):
        data.append(np.array(img))
        result.append(encoder.transform([[1]]).toarray())
```

```
[12] data = np.array(data)
data.shape
```

```
[13] result = np.array(result)
result = result.reshape(139,2)
```

```
[14] x_train,x_test,y_train,y_test = train_test_split(data, result, test_size=0.2,
shuffle=True, random_state=0)
```

```
[15] model = Sequential()
```

```
model.add(Conv2D(32, kernel_size=(2, 2), input_shape=(128, 128, 3), padding =
'Same'))
```

```
model.add(Conv2D(32, kernel_size=(2, 2), activation = 'relu', padding = 'Same'))
```

```
model.add(BatchNormalization())
```

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Dropout(0.25))
```

```
model.add(Conv2D(64, kernel_size = (2,2), activation = 'relu', padding = 'Same'))
```

```
model.add(Conv2D(64, kernel_size = (2,2), activation = 'relu', padding = 'Same'))
```

```
model.add(BatchNormalization())
```

```
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
```

```
model.add(Dropout(0.25))
```

```
model.add(Flatten())
```

```
model.add(Dense(512, activation='relu'))
```

```
model.add(Dropout(0.5))
```

```
model.add(Dense(2, activation='softmax'))
```

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

```
print(model.summary())
```

```
[16] y_train.shape
```

```
[17] history = model.fit(x_train, y_train, epochs = 30, batch_size = 40, verbose =
1, validation_data = (x_test, y_test))
```

```
[18] plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Test', 'Validation'], loc='upper right')
plt.show()
```

```
[19] plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Test', 'Validation'], loc='upper right')
plt.show()
```

```
[20] def names(number):
    if number==0:
        return 'Its a Tumor'
    else:
        return 'No, Its not a tumor'
```

```
[21] from matplotlib.pyplot import imshow
img = Image.open(r"/content/no/37 no.jpg")
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
res = model.predict_on_batch(x)
classification = np.where(res == np.amax(res))[1][0]
imshow(img)
print(str(res[0][classification]*100) + '% Confidence This Is ' + names(classification))
```

```
[22] from matplotlib.pyplot import imshow
img = Image.open(r"/content/yes/Y120.JPG")
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
res = model.predict_on_batch(x)
classification = np.where(res == np.amax(res))[1][0]
imshow(img)
print(str(res[0][classification]*100) + '% Confidence This Is A ' + names(classification))
```

5.2. PROJECT MODULES

These are the modules in our project:

- Image input
- Image pre-processing
- Splitting data to train and test
- Brain tumour image classification using CNN
- Checking model with getting Output

Module 1: Image input

- We have collected the data from the kaggle website for Brain Tumour Detection.
- The Dataset is taken with images.
- These images are splitted into two folders which are yes and no each containing images with and without brain tumours respectively.
- Our dataset is taken from kaggle.

Module 2: Image pre-processing

- What is Image pre-processing?

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it.

- Let's get into our project -

We have used one hot encoding in our project.

What is One Hot Encoding?

One hot encoding is a process of converting the categorical data variables to be provided to machine and deep learning algorithms which in turn improve predictions as well as classification accuracy of a model.

Why we have used one hot encoding process in our project?

As we have two outputs/classes which are yes and no. By using one hot encoder we are assigning variables to them.

The variables we assigned are 0 and 1.

0 - Having Tumour i.e. yes.

1 – No Tumour.

We are using one hot encoder for just reading whether the given input contains brain tumour or not.

Steps involved in pre-processing module of our project :

1. First step is Creating three important lists -
 - a. data list for storing image data in numpy array form
 - b. paths list for storing paths of all images
 - c. result list for storing one hot encoded form of target class whether normal or tumour
2. We append all the jpg files from both yes and no folders into the raw path.
3. Resizing of the images are done.
4. Images are converted into numpy array form.
5. These images are now appended to the data list.
6. Now append the result into the result path by using one hot encoder, by storing it as 0 or 1. After that data and the result paths are reshaped to get clean and pure data.

Module 3: Splitting the data to train and test

We split the dataset into training and testing sets using the `train_test_split()` function in the Scikit-learn package.

The test size we have taken is 0.2 (i.e. 20% for testing and 80% for training).

Random state is 0.

And zero shuffle is true.

Module 4: Brain tumour image classification using CNN (Model Building and fitting pre-processed data into model)

Classification is the best approaches for identification of images like any kind of medical imaging. All classification algorithms are based on the prediction of image, where one or more features and that each of these features belongs to one of several classes. An automatic and reliable classification method Convolutional Neural Network (CNN) will be used since it is robust in structure which helps in identifying every minute details. A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance to various aspects/objects in the image and be able to differentiate one from the other. The preprocessing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNet have the ability to learn these filters/characteristics. A ConvNet is able to successfully capture the spatial and temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better. The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. For this step we need to import Keras and other packages that we're going to use in building the CNN. Import the following packages:

- Sequential is used to initialize the neural network.
- Convolution2D is used to make the convolutional network that deals with the images.
- MaxPooling2D layer is used to add the pooling layers.
- Flatten is the function that converts the pooled feature map to a single column that is passed to the fully connected layer.
- Dense adds the fully connected layer to the neural network.

SEQUENTIAL:

- To initialize the neural network, we create an object of the Sequential class.

- `classifier = Sequential ()`

CONVOLUTION:

- To add the convolution layer, we call the add function with the classifier object and pass in Convolution2D with parameters. The first argument feature_detectors which is the number of feature detectors that we want to create. The second and third parameters are dimensions of the feature detector matrix.

- We used 256 feature detectors for CNNs. The next parameter is input shape which is the shape of the input image. The images will be converted into this shape during pre-processing. If the image is black and white it will be converted into a 2D array and if the image is coloured it will be converted into a 3D array.

- In this case, we'll assume that we are working with coloured images. Input_shape is passed in a tuple with the number of channels, which is 3 for a coloured image, and the dimensions of the 2D array in each channel. If you are not using a GPU it's advisable to use lower dimensions to reduce the computation time. The final parameter is the activation function. Classifying images is a nonlinear problem. So, we use the rectifier function to ensure that we don't have negative pixel values during computation. That's how we achieve non-linearity.

- `classifier.add (Convolution2D (256, 3, 3, input_shape = (256, 256, 3), activation='relu'))`

POOLING:

- The Pooling layer is responsible for reducing the spatial size of the convolved feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model.
- There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. Generally, we use max pooling.
- In this step we reduce the size of the feature map. Generally, we create a pool size of 2x2 for max pooling. This enables us to reduce the size of the feature map while not losing important image information.
- `classifier.add (MaxPooling2D (pool_size= (2,2)))`

FLATTENING:

- In this step, all the pooled feature maps are taken and put into a single vector for inputting it to the next layer.
- The Flatten function flattens all the feature maps into a single long column.
- `classifier.add (Flatten ())`

FULLY CONNECTION:

- The next step is to use the vector we obtained above as the input for the neural network by using the Dense function in Keras. The first parameter is output which is the number of nodes in the hidden layer. You can determine the most appropriate number through experimentation. The higher the number of dimensions the more computing resources you will need to fit the model. A common practice is to pick the number of nodes in powers of two.
- `classifier.add (Dense (output = 64))`
- The next layer we have to add is the output layer. In this case, we'll use the sigmoid activation function since we expect a binary outcome. If we expected more than two outcomes, we would use the SoftMax function.

- The output here is 1 since we just expect the predicted probabilities of the classes.
- `classifier.add(Dense (output=1, activation='sigmoid'))`
- Now we train our model.
- We plot the loss function.

Module 5: Checking the model to get output

- This will give the output when our image is given as input as percentage confidence along with information whether tumour is present or not.

6. SYSTEM TESTING

6.1. INTRODUCTION

Software Testing, The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product .It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2. TESTING METHODS

The following are the Testing Methodologies:

- Unit Testing.
- Integration Testing.
- Black Box Testing.
- White Box Testing.
- Validation Testing.

Unit Testing-

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing. All error handling paths are also tested to get expected result.

Integration Testing-

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design. The following are the types of Integration Testing:

1. Top Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module.

2. Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated.

Validation Checking-

Validation checks are performed on the following fields.

White Box Testing-

White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security. In white box testing, code is visible to testers so it is also called Clear box testing, open box testing, transparent box testing, Code-based testing and Glass box testing.

It is one of two parts of the Box Testing approach to software testing. Its counterpart, Black box testing, involves testing from an external or end-user type perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

Black Box Testing-

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioural Testing.

6.3. TEST CASES

While testing the web applications, one should consider the below mentioned template. The below mentioned checklist is almost applicable for all types of web applications depending on the business requirements.

The web application testing checklist consists of-

- Usability Testing
- Functional Testing
- Compatibility Testing
- Database Testing
- Security Testing

Usability Testing:

- Usability testing is nothing but the User-friendliness check.

Functional Testing:

- Testing the features and operational behavior of a product to ensure they correspond to its specifications.
- Testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions.

Compatibility Testing:

- Compatibility testing is used to determine if your software is compatible with other elements of a system with which it should operate, e.g. Browsers, Operating Systems, or hardware.
- The purpose of Compatibility testing is to evaluate how well software performs in a particular browser, Operating Systems, hardware or software.

Database Testing:

- In Database testing backend records are tested which have been inserted through the web or desktop applications. The data which is displaying in the web application should match with the data stored in the Database.

Security Testing:

- Security Testing involves the test to identify any flaws and gaps from a security point of view.

Performance Testing:

- Performance Testing is conducted to evaluate the compliance of a system or component with specified performance requirements.

Brain Tumor detection using CNN

```
from matplotlib.pyplot import imshow
img = Image.open(r"/content/no/37 no.jpg")
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
res = model.predict_on_batch(x)
classification = np.where(res == np.amax(res))[1][0]
imshow(img)
print(str(res[0][classification]*100) + '% Confidence This Is ' + names[classification])
```

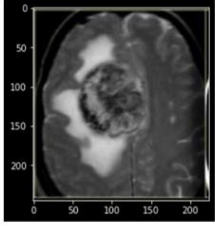
100.0% Confidence This Is No, Its not a tumor



(fig:The output is that it's not a tumour when a no tumour image is given)

```
from matplotlib.pyplot import imshow
img = Image.open(r"/content/yes/Y120.JPG")
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
res = model.predict_on_batch(x)
classification = np.where(res == np.amax(res))[1][0]
imshow(img)
print(str(res[0][classification]*100) + '% Confidence This Is A ' + names[classification])
```

83.04874897003174% Confidence This Is A Its a Tumor



(fig:The output is that it's a tumour when a tumour image is given)

7. CONCLUSION

The Project "Brain Tumour Detection Using CNN" successfully implemented all features and tested will all the possible test cases and achieved satisfying results. The main aim of this project is to be able improves the accuracy with minimal computational time and less complexity and achieve the validation accuracy will be high and validation loss will be very low, time taken for detecting the brain tumor is relatively less in comparison with the algorithms proposed earlier. The aim of this project is attained through CNN Model.

8. BIBLIOGRAPHY

BRAIN TUMOR IMAGES DATASET

<https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection>

OTHER REFERENCES

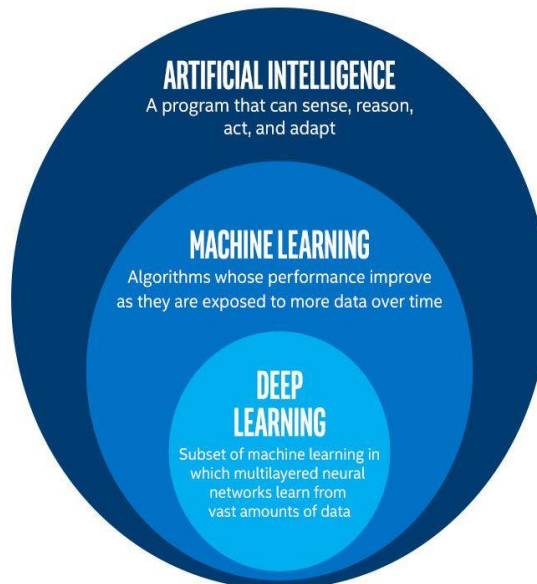
<https://www.mayoclinic.org/diseases-conditions/brain-tumor/diagnosis-treatment/drc-20350088>

<https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection>

9. APPENDIX

9.1. INTRODUCTION TO MACHINE LEARNING

Machine learning is a subset of Artificial Intelligence. Machine Learning consists of statistical methods that enable machines to improve with experience. It focuses mainly on the designing of systems, thereby allowing them to learn and make predictions based on experience.



Unsupervised Learning is a machine learning technique in which the users do not need to supervise the model. Instead, it allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with the unlabeled data.

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.

Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation.