BRAIN
Tumor Detection

A machine learning approach

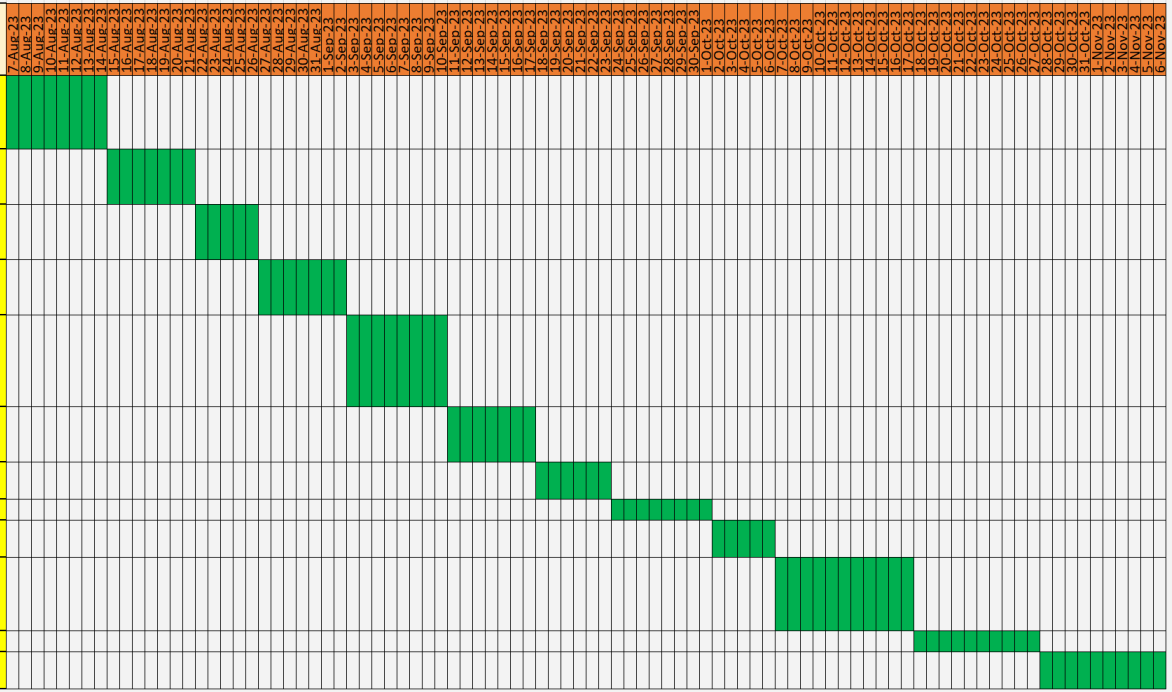
Group: 01

Mohammad Akib	2012574642
Istiak Ahasan	2012082042
Salman Farsi	2021702042
Tanvir Ahmed Chowdhury	2031227642

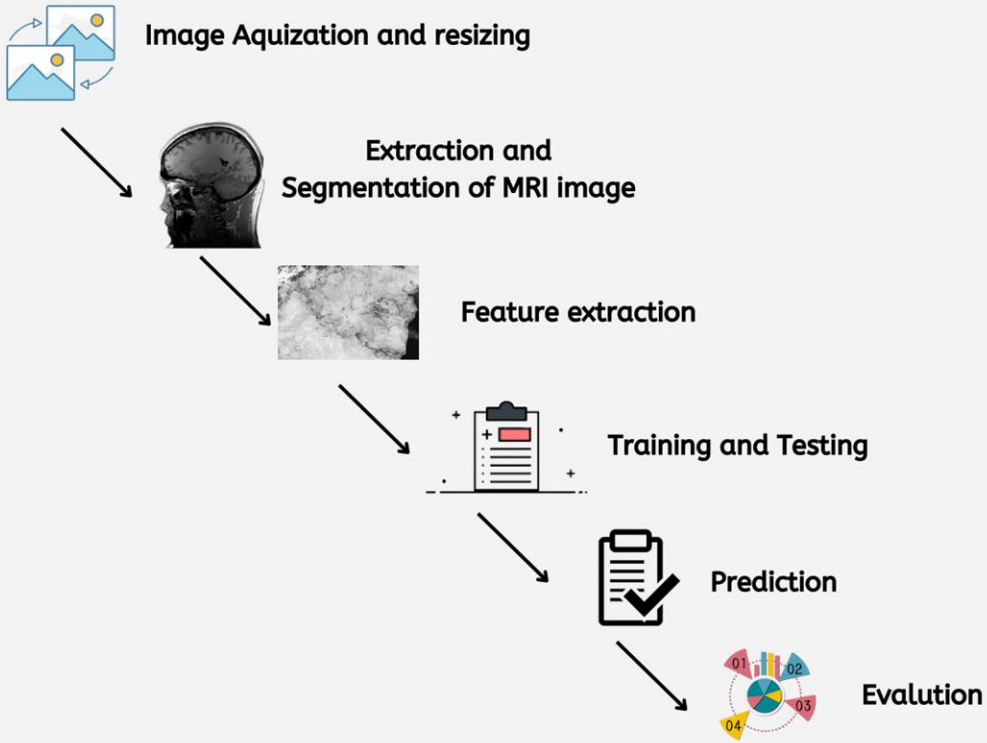
Gantt Chart

Project Name	Project Duration	Project Start Date	Project End Date
Brain Tumor Detection	91	7-Aug-23	6-Nov-23

Task ID	Task Description	Task Duration	Start Date	End Date
1	Read related researchpapers & have an idea of previous done works	7	7-Aug-23	14-Aug-23
2	Read some more research papers & search for newer datasets	6	15-Aug-23	21-Aug-23
3	Find some more datasets & see what they offer new	4	22-Aug-23	26-Aug-23
4	Have a concept about image processing and learn how they works	6	27-Aug-23	2-Sep-23
5	Learn more about image processing,preprocess dataset & apply them on Feature Extraction & Selection	7	3-Sep-23	10-Sep-23
6	Browse the already existing model we can find	6	11-Sep-23	17-Sep-23
7	Select the model we are going to continue with	5	18-Sep-23	23-Sep-23
8	Build Model	7	24-Sep-23	1-Oct-23
9	Test the model using test set	4	2-Oct-23	6-Oct-23
10	Model Tuning & optimization, PerformanceEvaluation & Analysis	10	7-Oct-23	17-Oct-23
11	Model Interpretation	9	18-Oct-23	27-Oct-23
12	Model Visualization & evaluate its accuracy	9	28-Oct-23	6-Nov-23



System Diagram



Introduction

Our brain tumor detection project aims to develop an advanced system that combines imaging techniques and machine learning algorithms to improve the accuracy and efficiency of brain tumor diagnosis.

- Our aim was to enhance the early detection and treatment of brain tumors, leading to improved patient outcomes.
- Detect brain tumor just by the clear picture of MRI scan with reduced time and greater accuracy.

Introduction

The newer, BR35H dataset provides a diverse collection of brain images, encompassing different types and grades of tumors, as well as patient demographics. This dataset serves as a valuable resource for training and validating our detection system, enabling us to develop robust algorithms.

With the integration of Convolutional Neural Networks (CNN), we seek to enhance the performance and effectiveness of brain tumor detection, ultimately benefiting healthcare professionals and patients alike.

Dataset

In our brain tumor detection project, we have utilized the BR35H dataset to develop an efficient system for accurate identification of brain tumors.

Has total 3000 Pre-processed MRI images.

Half of them are with brain tumors that are in a folder named ‘yes’

Half of them which are not, is in a folder labeled as ‘no’.

All of the images are skull striped.

Background

01

In most of the places, still segmentation part is done manually. It consumes more time and could end up with errors

02

In rural areas, though there might be centers to do the MRI but has doctor availibility issue.
Even doctors could cause human errors.

03

It takes too much time and money to stay on a line and show the report to a doctor.
It’s painful for elderlies and office going busy persons.

Background

So, to initiate our background study for the brain tumor detection project, we have explored the background of neural networks and review relevant research papers on the same topic. We tried to understand the foundational concepts of neural networks and examining existing studies that provided valuable insights into the application of neural networks, specifically Convolutional Neural Networks (CNNs), in the context of brain tumor detection.

Why CNN

The architecture of a CNN is inspired by the organization of the visual cortex in the human brain. It consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers.

- The convolutional layers use filters to scan the input image and detect relevant features through the process of convolution.
- The pooling layers down-sample the spatial dimensions of the feature maps, reducing computational complexity while retaining important information.
- Finally, the fully connected layers aggregate the extracted features and make predictions based on them.

Why CNN

One of the key advantages of CNN is their ability to capture spatial hierarchies of features.

The early layers of the network learn basic features such as edges and textures, while deeper layers learn more complex and abstract features.

This hierarchical representation allows CNNs to effectively model complex patterns in images.

Methodology

After acquiring the images from our Br35H dataset, we down-size the original image from 256 × 256 to 64 × 64 .

Then we transformed all the images into NumPy arrays (available in python) so that our model can take up less space.

Before splitting the dataset, we have shuffled the data so that our model can train on unordered data. After shuffling the data, we divide the dataset into three sections including train, test, and validation.

Approximately 80% of the data is used for training, and a further 20% is used for validation and testing purposes

Here we tried CNN approach to build the model for our work.

While implementing CNN, for the first Conv2D layers as parameters we used how many filters we want to use and then the kernel size. Then the ReLU activation comes and after that the image shape function.
In image shape function there are 3 parameters, first two is the shape of the image, and here third digit 3 refers to it's image type, which is RGB here.

Same goes for other functions here.

Code:

```
model=Sequential([

                    #implementing cnn
                    layers.Conv2D(50,(3,3),
                                  activation="relu",
                                  input_shape=(64,64,3)),
                    layers.MaxPooling2D((2,2)),

                    layers.Conv2D(64,(3,3),
                                  activation="relu"),
                    layers.MaxPooling2D((2,2)),

                    layers.Conv2D(54,(3,3),
                                  activation="relu"),
                    layers.MaxPooling2D((2,2)),

                     #dense_layer implementation
                     layers.Flatten(),
                     layers.Dense(64,activation="relu"),
                     layers.Dense(2,activation="softmax")



])

model.summary()
```

This shows the layering techniques and reduction of output sizes through every layer approach. Here's Param represents parameter that indicates the combine values of weights and biases.

Output:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 50)        1400

 max_pooling2d (MaxPooling2D  (None, 31, 31, 50)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 64)        28864

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 12, 12, 54)        31158

 max_pooling2d_2 (MaxPooling  (None, 6, 6, 54)         0
 2D)

 flatten (Flatten)           (None, 1944)              0

 dense (Dense)               (None, 64)                124480

 dense_1 (Dense)             (None, 2)                 130

=================================================================
Total params: 186,032
Trainable params: 186,032
Non-trainable params: 0
_________________________________________________________________
```

Here, we are showing the gradual increment of accuracy in every epoch. It rose from 98.7% to all the way 99.4%.

We can still see some ups and downs, but there's no such massive drop in accuracy and spike in loss section as there was before.

```
[18] model.compile(loss="sparse_categorical_crossentropy", metrics=['accuracy'])
     model.fit(X_train,y_train,epochs=16)

     Epoch 1/16
     75/75 [==============================] - 21s 269ms/step - loss: 0.0741 - accuracy: 0.9871
     Epoch 2/16
     75/75 [==============================] - 21s 286ms/step - loss: 0.0479 - accuracy: 0.9921
     Epoch 3/16
     75/75 [==============================] - 20s 271ms/step - loss: 0.0169 - accuracy: 0.9954
     Epoch 4/16
     75/75 [==============================] - 21s 279ms/step - loss: 0.0246 - accuracy: 0.9954
     Epoch 5/16
     75/75 [==============================] - 21s 285ms/step - loss: 0.0181 - accuracy: 0.9962
     Epoch 6/16
     75/75 [==============================] - 20s 266ms/step - loss: 0.0194 - accuracy: 0.9967
     Epoch 7/16
     75/75 [==============================] - 22s 288ms/step - loss: 0.0395 - accuracy: 0.9908
     Epoch 8/16
     75/75 [==============================] - 21s 286ms/step - loss: 0.0100 - accuracy: 0.9971
     Epoch 9/16
     75/75 [==============================] - 20s 268ms/step - loss: 0.0449 - accuracy: 0.9946
     Epoch 10/16
     75/75 [==============================] - 22s 289ms/step - loss: 0.0010 - accuracy: 0.9996
     Epoch 11/16
     75/75 [==============================] - 21s 276ms/step - loss: 0.0225 - accuracy: 0.9950
     Epoch 12/16
     75/75 [==============================] - 21s 275ms/step - loss: 0.0239 - accuracy: 0.9958
     Epoch 13/16
     75/75 [==============================] - 22s 288ms/step - loss: 0.0219 - accuracy: 0.9962
     Epoch 14/16
     75/75 [==============================] - 20s 269ms/step - loss: 0.0289 - accuracy: 0.9962
     Epoch 15/16
     75/75 [==============================] - 22s 291ms/step - loss: 0.0794 - accuracy: 0.9929
     Epoch 16/16
     75/75 [==============================] - 22s 291ms/step - loss: 0.0351 - accuracy: 0.9942
     <keras.callbacks.History at 0x7f859c8eb340>
```

Classification Report

This classification report is our performance evaluation matrix.

Here, Precision is defined as the ratio of true positives to the sum of true and false positives.
Recall is defined as the ratio of true positives to the sum of true positives and false negatives.

The F1 is the mean of precision and recall. The closer the value of the F1 score is to 1.0, the better the expected performance of the model is.
Macro Average is averaging the total true positives, false negatives and false positives.

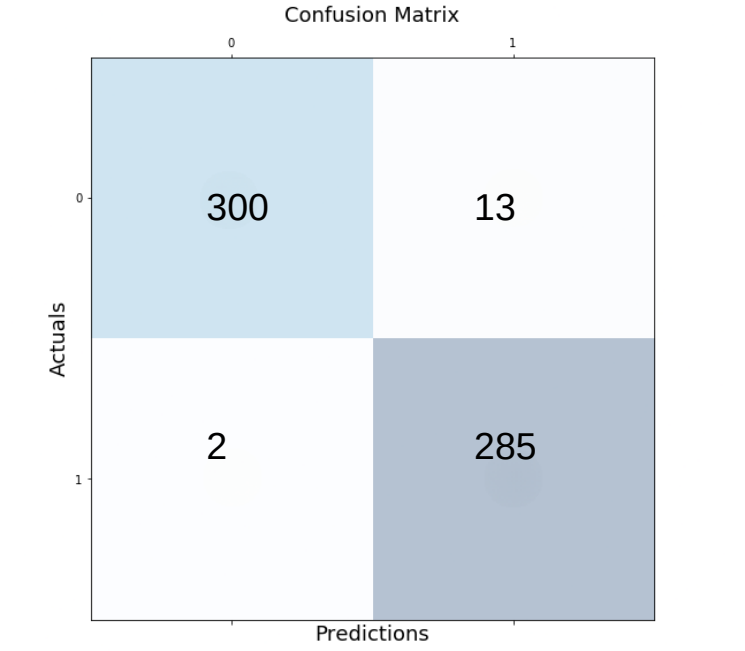
And lastly, Support is the number of actual occurrences of the class in the dataset.

	precision	recall	f1-score	support
0	0.99	0.96	0.98	313
1	0.96	0.99	0.97	287
accuracy			0.98	600
macro avg	0.97	0.98	0.98	600
weighted avg	0.98	0.97	0.98	600

Confusion Matrix

We can see that we got 300 True-Positive, 13 False-Positive, 2 False-Negetive, 285 True-Negetive.

That means, in 300 cases we predicted Yes and they do have brain tumor.
In 13 cases we predicted Yes, but they are healthy brain.
In in 2 cases we predicted No and they don't have the tumor.
And in the rest, we predicted False, and they actually don't have the tumor.




```
[[300  13]
 [  2 285]]
```

We tried to go through some papers to compare the accuracy they have with our proposed model that used Br35H dataset.
As we have showed before, we do have a accuracy of 98%.
As Br35H is a newer dataset, so it is tough to find the available papers for free. But we did find some Open Access papers that used Br35H as a dataset.

We have included 3 of such that fortunately used the same dataset and the ratio.
They used 20% for the testing purpose and the rest for training purpose.

Authors	Algorithm	Accuracy
Gómez-Guzmán et al.	InceptionV3	97.1%
Naseer et al.	CNN-based CAD	97.8%
Raza et al.	GoogleNet	99.67%
Model we proposed	CNN	98.0%

Work Plan

Week Number	Work Flow
01	Read related research papers and have a idea about previously done works
02	Read more research papers, Search for a newer data set, and see what features they provide.
03	Have a concept about image processing and learn how they works.
04	Learn more about Image Processing, preprocess dataset and apply them on Feature Extraction and Selection
05	Browse the already existing models, select the model we are going to continue with
06	Build model
07	Test the model using test set
08	Model Tuning and Optimization, Performance Evaluation and Analysis
09	Model Interpretation, Visualization and evaluate it’s accuracy.

Conclusion

The goal of detecting brain tumors requires high levels of sensitivity, specificity, and accuracy. If these outcomes are obtained, it will help in the early diagnosis of brain tumors, and doctors will be able to spend more time treating patients rather than carrying out such tiresome activities as detecting brain tumors.

Till now we have created a mere basic CNN model to detect the existence of brain tumor in MR images.
Modification of this model furthermore can aquire more accuracy from that available dataset.

It is highly challenging for the model to be able to distinguish between the many structures present in an image because of the poor contrast in the source photos.
Hopefully, we will be continuing this work and modifying and creating a model to gain higher accuracy with the dataset we are using. We also intend to continue this work in the future in order to gain knowledge and create a special solution that will produce superior outcomes.

THANK YOU !

For Listening