

INT247 Machine Learning Foundation

Class Assignment - 1



LOVELY
PROFESSIONAL
UNIVERSITY

Topic - Brain Tumor Prediction

Name : Anubhav Joshi

Registration Number : 11907702

Section : KM015

Roll Number : 11

Date of Submission : 30/03/2022

ACKNOWLEDGEMENT

The success and final outcome of learning Machine Learning required a lot of guidance and assistance from many people, and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I am thankful to and fortunate enough to get constant encouragement, support and guidance from Dr Sager Pandey and my seniors which helped me in successfully completing my project work.

(Signature of Student)

Name of Student - Anubhav Joshi

Registration Number - 11907702

Date : 30-03-2022

Annexure-II : Student Declaration

To whomsoever it may concern

I, Anubhav Joshi, Registration Number : 11907702, hereby declare that the work done by me on “**Brain Tumor Prediction**” from **January, 2022 to March, 2022** is a record of original work for the partial fulfilment of requirements for the awards of the degree, B.Tech CSE.

Anubhav Joshi (11907702)

Signature of the student

Dated: 30-03-2022

INDEX

1. Abstract
2. Introduction
3. Overview of the Project
4. Software Tools
5. Result
6. Conclusion
7. Reference

Abstract

- A Brain tumor is considered as one of the aggressive diseases, among children and adults. Brain tumors account for 85 to 90 percent of all primary Central Nervous System(CNS) tumors. Every year, around 11,700 people are diagnosed with a brain tumor. The 5-year survival rate for people with a cancerous brain or CNS tumor is approximately 34 percent for men and 36 percent for women. Brain Tumors are classified as: Benign Tumor, Malignant Tumor, Pituitary Tumor, etc. 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. These images are examined by the radiologist. A manual examination can be error-prone due to the level of complexities involved in brain tumors and their properties.
- Application of automated classification techniques using Machine Learning(ML) and Artificial Intelligence(AI) has consistently shown higher accuracy than manual classification. Hence, proposing 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.
- Brain Tumors are complex. There are a lot of abnormalities in the sizes and location of the brain tumor(s). This makes it really difficult for complete understanding of the nature of the tumor. Also, a professional Neurosurgeon is required for MRI analysis. Often times in developing countries the lack of skillful doctors and lack of knowledge about tumors makes it really challenging and time-consuming to generate reports from MRI. So an automated system can solve this problem.

INTRODUCTION

The central nervous system disseminates sensory information and its corresponding actions throughout the body. The brain, along with the spinal cord, assists in this dissemination. The brain's anatomy contains three main parts; brain stem, cerebrum, and cerebellum. The weight of a normal human brain is approximately 1.2–1.4 K, with a volume of 1260 cm³ (male brain) and 1130 cm³ (female brain). The frontal lobe of the brain assists in problem-solving, motor control, and judgments.

The parietal lobe manages body position. The temporal lobe controls memory and hearing functions, and occipital lobe supervises the brain's visual processing activities. The outer part of the cerebrum is known as cerebral cortex, and is a greyish material; it is composed of cortical neurons. The cerebellum is relatively smaller than the cerebrum. It is responsible for motor control, i.e., systematic regulation of voluntary movements in living organisms with a nervous system. Due to variable size and stroke territory, ALI, lesionGnb, and LINDA methods fail to detect the small lesion region. Cerebellum is well-structured and well-developed in human beings as compared to other species.

The cerebellum has three lobes; an anterior, a posteriori, and a flocculonodular. A round-shaped structure named vermis connects the anterior and posterior lobes. The cerebellum consists of an inner area of white matter (WM) and an outer greyish cortex, which is a bit thinner than that of the cerebrum. The anterior and posterior lobes assist in the coordination of complex motor movements. The flocculonodular lobe maintains the body's balance. The brain stem, as the name states, is a 7–10 cm-long stem-like structure. It contains cranial and peripheral nerve bundles and assists in eye movements and regulations, balance and maintenance, and some essential activities such as breathing. The nerve tracks originating from the cerebrum's thalamus pass through the brainstem to reach the spinal cord. From there, they spread throughout the body. The main parts of the brainstem are midbrain, pons, and medulla. The midbrain assists in functions such as motor, auditory, and visual processing, as well as eye movements. The pons assists in breathing, intra-brain communication, and sensations, and medulla oblongata helps in blood regulation, swallowing, sneezing, etc.

Brain tumor and stroke lesions

Brain tumors are graded as slow-growing or aggressive . A benign (slow-growing) tumor does not invade the neighbouring tissues; in contrast, a malignant (aggressive) tumor propagates itself from an initial site to a secondary site . According to WHO, a brain tumor is categorised into grades I–IV. Grades I and II tumors are considered as slow-growing, whereas grades III and IV tumors are more aggressive, and have a poorer prognosis . In this regard, the details of brain tumor grades are as follows.

Grade I: These tumors grow slowly and do not spread rapidly. These are associated with better odds for long-term survival and can be removed almost completely by surgery. An example of such a tumor is grade 1 pilocytic astrocytoma.

Grade II: These tumors also grow slowly but can spread to neighbouring tissues and become higher grade tumors. These tumors can even come back after surgery. Oligodendroglioma is a case of such a tumor.

Grade III: These tumors develop at a faster rate than grade II, and can invade the neighbouring tissues. Surgery alone is insufficient for such tumors, and post-surgical radiotherapy or chemotherapy is recommended. An example of such a tumor is anaplastic astrocytoma.

Grade IV: These tumors are the most aggressive and are highly spreadable. They may even use blood vessels for rapid growth. Glioblastoma multiforme is such a type of tumor .

Ischemic stroke: Ischemic stroke is an aggressive disease of the brain and it is a major cause of disability and death around the globe . An ischemic stroke occurs when the blood supply to the brain is cut off, resulting underperfusion (in tissue hypoxia) and death of the advanced tissues in hours. Based on the severity, stroke lesions are categories into different stages such as acute (0–24 h), sub-acute (24 h–2 weeks) and chronic (> 2 weeks) .

OVERVIEW OF THE PROJECT

- Connecting to Google Drive for

```
✓ [1] from google.colab import drive
27s drive.mount('/content/drive')

Mounted at /content/drive

✓ [2] import os
0s Root = "/content/drive/MyDrive/BrainTumorDataset"
os.chdir(Root)
```

- Importing Libraries

```
✓ [3] #Import the necessary libraries first
3s import tensorflow as tf
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from tensorflow.keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.metrics import categorical_crossentropy
from keras.models import Sequential, Model
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
from keras.layers import Activation, Dropout, BatchNormalization, Flatten, Dense, AvgPool2D, MaxPool2D
from keras.models import Sequential, Model
from tensorflow.keras.optimizers import Adam
import cv2
```


- Importing the dataset

```

✓ [4] data = '/content/drive/MyDrive/BrainTumorDataset/'
0s No_brain_tumor = '/content/drive/MyDrive/BrainTumorDataset/datasets/no/'
    Yes_brain_tumor = '/content/drive/MyDrive/BrainTumorDataset/datasets/yes/'

✓ [5] dirlist=[No_brain_tumor, Yes_brain_tumor]
11s classes=['No', 'Yes']
    filepaths=[]
    labels=[]
    for i,j in zip(dirlist, classes):
        filelist=os.listdir(i)
        for f in filelist:
            filepath=os.path.join (i,f)
            filepaths.append(filepath)
            labels.append(j)
    print ('filepaths: ', len(filepaths), ' labels: ', len(labels))

filepaths: 3000 labels: 3000

```

- Visualise the dataset

```

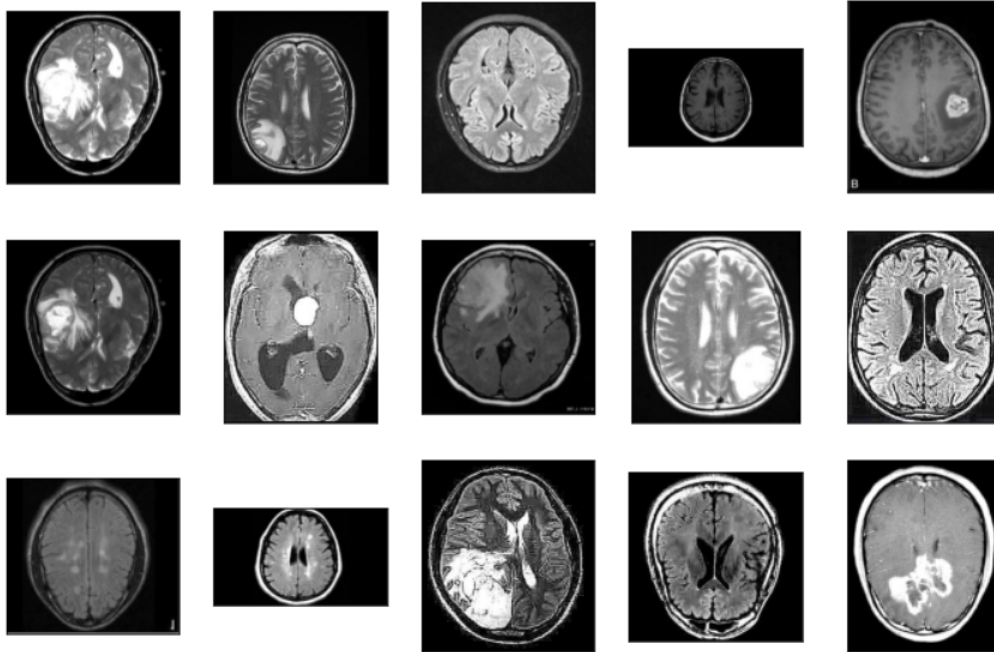
✓ [8] #visualize brain tumor images
6s

plt.figure(figsize=(12,8))
for i in range(15):
    random = np.random.randint(1,len(df))
    plt.subplot(3,5,i+1)
    plt.imshow(cv2.imread(df.loc[random,"filepaths"]))
    plt.title(df.loc[random, "labels"], size = 15, color = "white")
    plt.xticks([])
    plt.yticks([])

plt.show()

```

```
plt.show()
```



- Splitting into train and test sets

✓
0s

```
[9] from sklearn.model_selection import train_test_split

train, test = train_test_split(df, train_size=0.95, random_state=0)
train_new, valid = train_test_split(train, train_size=0.90, random_state=0)

print(f"train set shape: {train_new.shape}")
print(f"test set shape: {test.shape}")
print(f"validation set shape: {valid.shape}")
```

```
train set shape: (2565, 2)
test set shape: (150, 2)
validation set shape: (285, 2)
```

- **Data Augmentation**

Let's start the modelling task The ImageDataGenerator for keras is awesome. It lets you augment your images in real-time while your model is still training! You can apply any random transformations on each training image as it is passed to the model. This will not only make your model robust but will also save up on the overhead memory! We will apply the Image Data Generator on training with various parameters, but we won't apply the same parameters on testing. Why? Because we want the test images as it is, we don't want biasedness, also if we fit it we will be applying the model only on these test images only, it can't predict new images if fed into the model Because new images will not be augmented this way.

```
✓ [10] train_datagen = ImageDataGenerator(rescale = 1./255.,rotation_range = 40, width_shift_range = 0.2, height_shift_range = 0.2,
0s shear_range = 0.2, zoom_range = 0.2, horizontal_flip = True, vertical_flip =True)
test_datagen = ImageDataGenerator(rescale = 1.0/255.)
```

Now fit them to get the images from directory (name of the images are given in dataframe) with augmentation

```
✓ [11] train_gen = train_datagen.flow_from_dataframe(dataframe = train_new,
0s x_col = 'filepaths', y_col = 'labels',
target_size = (224,224), batch_size = 32,
class_mode = 'binary', shuffle = True)

val_gen = train_datagen.flow_from_dataframe(valid,
target_size=(224,224), x_col = 'filepaths', y_col = 'labels',
class_mode='binary',
batch_size= 16, shuffle=True)

test_gen = test_datagen.flow_from_dataframe(test,
target_size = (224,224), x_col = 'filepaths', y_col = 'labels',
class_mode = 'binary',
batch_size = 16, shuffle = False)
```

Found 2565 validated image filenames belonging to 2 classes.
Found 285 validated image filenames belonging to 2 classes.
Found 150 validated image filenames belonging to 2 classes.

- **Choosing our Model**

Our base model is InceptionResNetV2, which is a pre-trained model.

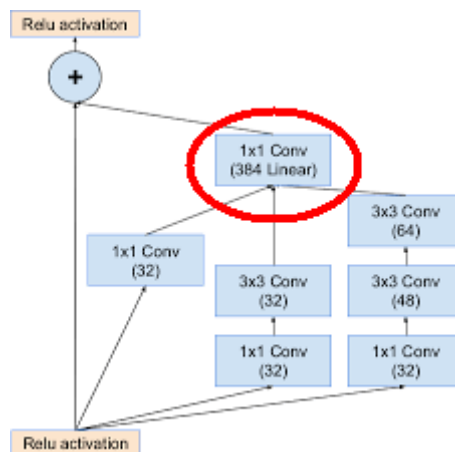
What is a pre-trained model?

A pre-trained model has been previously trained on a dataset and contains the weights and biases that represent the features of whichever dataset it was trained on. Learned features are often transferable to different data. For example, a model trained on a large dataset of bird images will contain learned features like edges or horizontal lines that would be transferable to your dataset.

ResNet and Inception have been central to the largest advances in image recognition performance in recent years, with very good performance at a relatively low computational cost.

Inception-ResNet combines the Inception architecture, with residual connections.

Residual Inception blocks

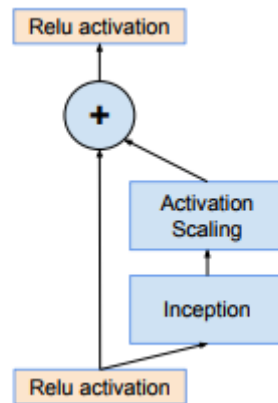


Residual Inception Block(Inception-ResNet-A)

1. Each Inception block is followed by a filter expansion layer (1×1 convolution without activation) which is used for scaling up the dimensionality of the filter bank before the addition to match the depth of the input.
2. In the case of Inception-ResNet, batch-normalisation is used only on top of the traditional layers, but not on top of the summations.

Scaling of Residuals

If the number of filters exceeded 1000, the residual variants started to exhibit instabilities and the network had just “died” early in the training, meaning that the last layer before the average pooling started to produce only zeros after a few tens of thousands of iterations. This could not be prevented, neither by lowering the learning rate nor by adding an extra batch normalisation to this layer.



Scaling of Residuals

According to them, scaling down the residuals before adding them to the previous layer activation seemed to stabilise the training. To scale the residuals, scaling factors between 0.1 and 0.3 were picked.

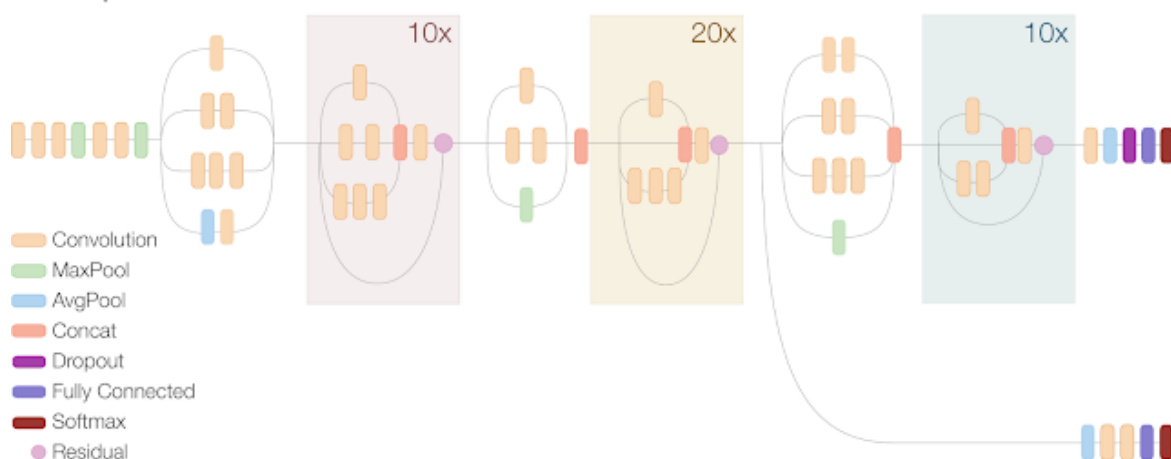
Inception-ResNet-v2

Inception-ResNet-v2 is a convolutional neural network that is trained on more than a million images from the ImageNet database. The network is 164 layers deep and can classify images into 1000 object categories, such as the keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 299-by-299, and the output is a list of estimated class probabilities.

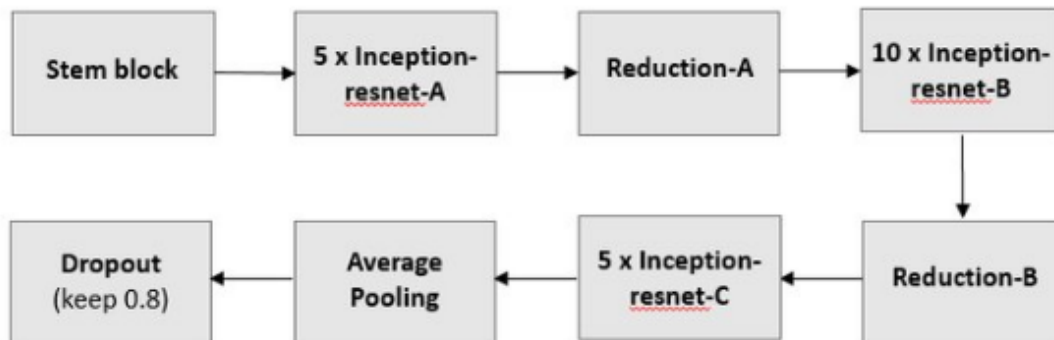
Inception Resnet V2 Network



Compressed View



It is formulated based on a combination of the Inception structure and the Residual connection. In the Inception-Resnet block, multiple sized convolutional filters are combined with residual connections. The usage of residual connections not only avoids the degradation problem caused by deep structures but also reduces the training time. The figure shows the basic network architecture of Inception-Resnet-v2.



The basic architecture of Inception-Resnet-v2.

- My implementation of Inception-Resnet-V2

```

3s ✓ from tensorflow import keras
base_model = keras.applications.ResNet50V2(
    weights="imagenet", # Load weights pre-trained on ImageNet.
    input_shape=(224, 224, 3),
    include_top=False,
) # Do not include the ImageNet classifier at the top.

# Freeze the base_model
base_model.trainable = False

# Create new model on top
inputs = keras.Input(shape=(224, 224, 3))

# The base model contains batchnorm layers. We want to keep them in inference mode
# when we unfreeze the base model for fine-tuning, so we make sure that the
# base_model is running in inference mode here.
x = base_model(inputs, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dropout(0.2)(x) # Regularize with dropout
outputs = keras.layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
  
```

```
model.summary()
```

```

[ ] Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/
94674944/94668760 [=====] - 0s 0us/step
94683136/94668760 [=====] - 0s 0us/step
Model: "model"

```

Layer (type)	Output Shape	Param #
=====		
input_2 (InputLayer)	[(None, 224, 224, 3)]	0
resnet50v2 (Functional)	(None, 7, 7, 2048)	23564800
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dropout (Dropout)	(None, 2048)	0
dense (Dense)	(None, 1)	2049

```

=====
Total params: 23,566,849
Trainable params: 2,049
Non-trainable params: 23,564,800

```

- **Compiling the model**

```

[ ] callbacks = [
    tf.keras.callbacks.ModelCheckpoint("Tumor_classifier_model_epochs10.h5", save_best_only=True, verbose = 0)
]

```

```
model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate= 0.0001), metrics=['accuracy'])
```

- **Model fitting**

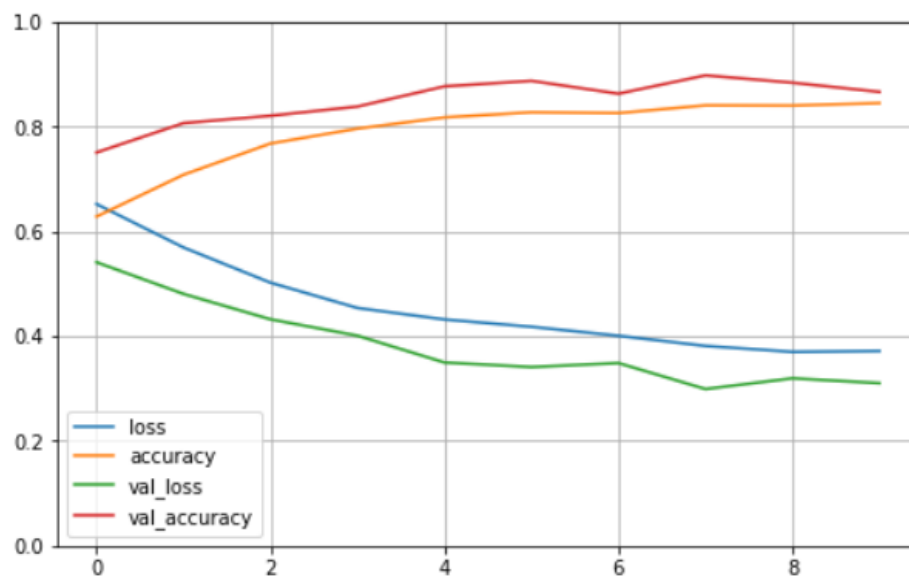
```
[ ] history = model.fit(train_gen, validation_data = val_gen, epochs = 10,
                        callbacks = [callbacks], verbose = 1)
```

```
Epoch 1/10
81/81 [=====] - 773s 9s/step - loss: 0.6525 - accuracy: 0.6285 - val_loss: 0.5414 - val_accuracy: 0.7509
Epoch 2/10
81/81 [=====] - 409s 5s/step - loss: 0.5695 - accuracy: 0.7084 - val_loss: 0.4809 - val_accuracy: 0.8070
Epoch 3/10
81/81 [=====] - 411s 5s/step - loss: 0.5023 - accuracy: 0.7680 - val_loss: 0.4323 - val_accuracy: 0.8211
Epoch 4/10
81/81 [=====] - 411s 5s/step - loss: 0.4540 - accuracy: 0.7965 - val_loss: 0.4010 - val_accuracy: 0.8386
Epoch 5/10
81/81 [=====] - 414s 5s/step - loss: 0.4320 - accuracy: 0.8179 - val_loss: 0.3498 - val_accuracy: 0.8772
Epoch 6/10
81/81 [=====] - 410s 5s/step - loss: 0.4180 - accuracy: 0.8277 - val_loss: 0.3412 - val_accuracy: 0.8877
Epoch 7/10
81/81 [=====] - 407s 5s/step - loss: 0.4009 - accuracy: 0.8261 - val_loss: 0.3487 - val_accuracy: 0.8632
Epoch 8/10
81/81 [=====] - 405s 5s/step - loss: 0.3815 - accuracy: 0.8409 - val_loss: 0.2992 - val_accuracy: 0.8982
Epoch 9/10
81/81 [=====] - 403s 5s/step - loss: 0.3702 - accuracy: 0.8405 - val_loss: 0.3196 - val_accuracy: 0.8842
Epoch 10/10
81/81 [=====] - 407s 5s/step - loss: 0.3719 - accuracy: 0.8452 - val_loss: 0.3104 - val_accuracy: 0.8667
```

```
[ ] model.save("/content/drive/MyDrive/BrainTumorDataset/Tumor_classifier_model.h5")
```

- **Accuracy of the Model**

```
[ ] pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



- **Prediction Time**

Here I choose an image from our dataset to predict if it contains a tumor or not, as you can see in the image address it was an image from the dataset containing images without the tumor.

After running this, the output was 0 which means the image we provided does not contain tumor, which is absolutely correct.

```
[ ] from PIL import Image
    model_path = "Tumor_classifier_model.h5"
    loaded_model = tf.keras.models.load_model(model_path)

    import matplotlib.pyplot as plt
    import numpy as np

    image = cv2.imread("/content/drive/MyDrive/BrainTumorDataset/datasets/no/No19.jpg")

    image_fromarray = Image.fromarray(image, 'RGB')
    resize_image = image_fromarray.resize((224, 224))
    expand_input = np.expand_dims(resize_image,axis=0)
    input_data = np.array(expand_input)
    input_data = input_data/255

    pred = loaded_model.predict(input_data)
    result = pred.argmax()
    result
```

0

SOFTWARE TOOLS

- **Google Colaboratory**

What is Google Colab?

Google Colab was developed by Google to provide free access to GPU's and TPU's to anyone who needs them to build a machine learning or deep learning model. Google Colab can be defined as an improved version of Jupyter Notebook.

What is a Jupyter Notebook?

Jupyter Notebook is an application that allows editing and running Notebook documents through a web browser or an Integrated Development Environment (IDE). Instead of files, you will work with Notebooks.

What is a Notebook?

Programming Languages are an intermediate form between human-understandable language and machine understandable language. Every application is built using one of the many programming languages available. Maybe a person with a computer science background can understand, but not everyone can. Remember, as Software Developers, we develop applications for people with little computer science knowledge.

Consider you are creating a machine learning model to improve customer satisfaction for a local store, in that case you will have to explain how the model can do this task, and you can't just explain him with your code base. Most people facing this situation will prepare a separate presentation. Notebooks were created so that it is not necessary. Notebook documents can include executable lines of code along with text, images, figures, tables, graphs, equations, and much more graphical data. In simple words, Notebook documents are a way of creating human-readable executable documents.

Google Colab Features

Google Colab provides tons of exciting features that any modern IDE offers, and much more. Some of the most exciting features are listed below.

- Interactive tutorials to learn machine learning and neural networks.
- Write and execute Python 3 code without having a local setup.
- Execute terminal commands from the Notebook.
- Import datasets from external sources such as Kaggle.
- Save your Notebooks to Google Drive.
- Import Notebooks from Google Drive.
- Free cloud service, GPUs and TPUs.
- Integrate with PyTorch, Tensor Flow, Open CV.
- Import or publish directly from/to GitHub.

RESULT

As mentioned earlier, the accuracy of the Inception Resnet V2 model here was near about 85% to 90% (approx) which is pretty decent considering the size of our dataset.

After a comprehensive review of the state-of-the-art exiting methods, the following challenges are found:

- The size of a brain tumor grows rapidly. Therefore, tumor diagnosis at an initial stage is an exigent task.
- Brain tumor segmentation is difficult owing to the following factors.
- MRI image owing to magnetic field fluctuations in the coil.
- Gliomas are infiltrative, owing to fuzzy borders. Thus, they become more difficult to segment.
- Stroke lesion segmentation is a very intricate task, as stroke lesions appear in complex shapes and with ambiguous boundaries and intensity variations.
- The optimised and best feature extraction and selection is another difficult process of inaccurate classification of brain tumors.


CONCLUSION

The accurate brain tumor detection is still very demanding because of tumor appearance, variable size, shape, and structure. Although tumor segmentation methods have shown high potential in analysing and detecting the tumor in MR images, still many improvements are required to accurately segment and classify the tumor region. Existing work has limitations and challenges for identifying substructures of tumor region and classification of healthy and unhealthy images.

In short, this survey covers all important aspects and latest work done so far with their limitations and challenges. It will be helpful for the researchers to develop an understanding of doing new research in a short time and correct direction.

The deep learning methods have contributed significantly but still require a generic technique. These methods provided better results when training and testing are performed on similar acquisition characteristics (intensity range and resolution); however, a slight variation in the training and testing images directly affects the robustness of the methods. In future work, research can be conducted to detect brain tumors more accurately, using real patient data from any medium (different image acquisition (scanners)). Handcrafted and deep features can be fused to improve the classification results. Similarly, lightweight methods such as quantum machine learning play a significant role to improve the accuracy and efficacy that save the time of radiologists and increase the survival rate of patients.

REFERENCES

-  Brain Tumor Detection using Machine Learning
- Dataset from Kaggle
<https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-detection>
- <https://link.springer.com/article/10.1007/s40747-021-00563-y>
- <https://www.scaler.com/topics/what-is-google-colab/>
- <https://medium.com/@zahraelhamraoui1997/inceptionresnetv2-simple-introduction-9a2000edcdb6>