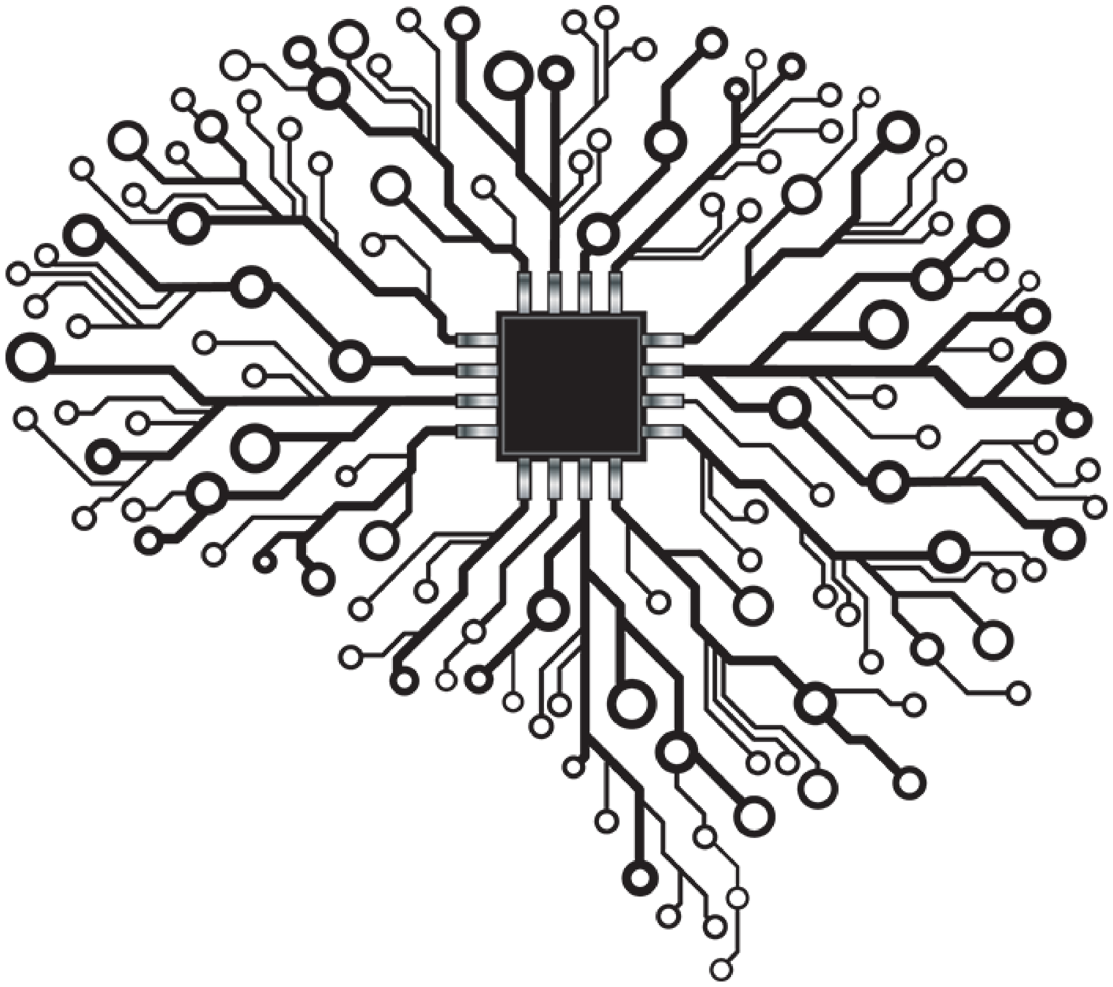
IDC-302 Project

Deep Learning based Brain Tumors Detection



Submitted by
Name- Priyank Dubey
Roll-no 19094
Department of Physical Sciences, IISER Berhampur

Overview

The brain is the most important organ in the human body, responsible for controlling and regulating all critical life functions for the body and a tumor is a mass of tissue formed by the accumulation of abnormal cells, which keep on growing. A brain tumor is a tumor which is either formed in the brain or has migrated No primary cause has been identified for the formation of tumors in the brain till date. Though tumors in the brain are not very common (Worldwide brain tumors make up only 1.8% of total reported tumors), the mortality rate of malignant brain tumors is very high due to the fact that the tumor formation is in the most critical organ of the body. Hence, it is of utmost importance to accurately detect brain tumors at early stages to lower the mortality rate.

Objective

The main purpose of this project was to build a deep neural network based Convolutional Neural Networks (CNN) model that would classify if subject has a tumor or not based on the images produced by Magnetic Resonance Imaging (MRI) scan. Since the tumor is very difficult to be seen via naked eyes. Hence, using deep learning will be feasible solution for the presented problem.

Deep Learning

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It’s achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

Convolutional Neural Network (CNN)

A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlaps to cover the entire visual area.

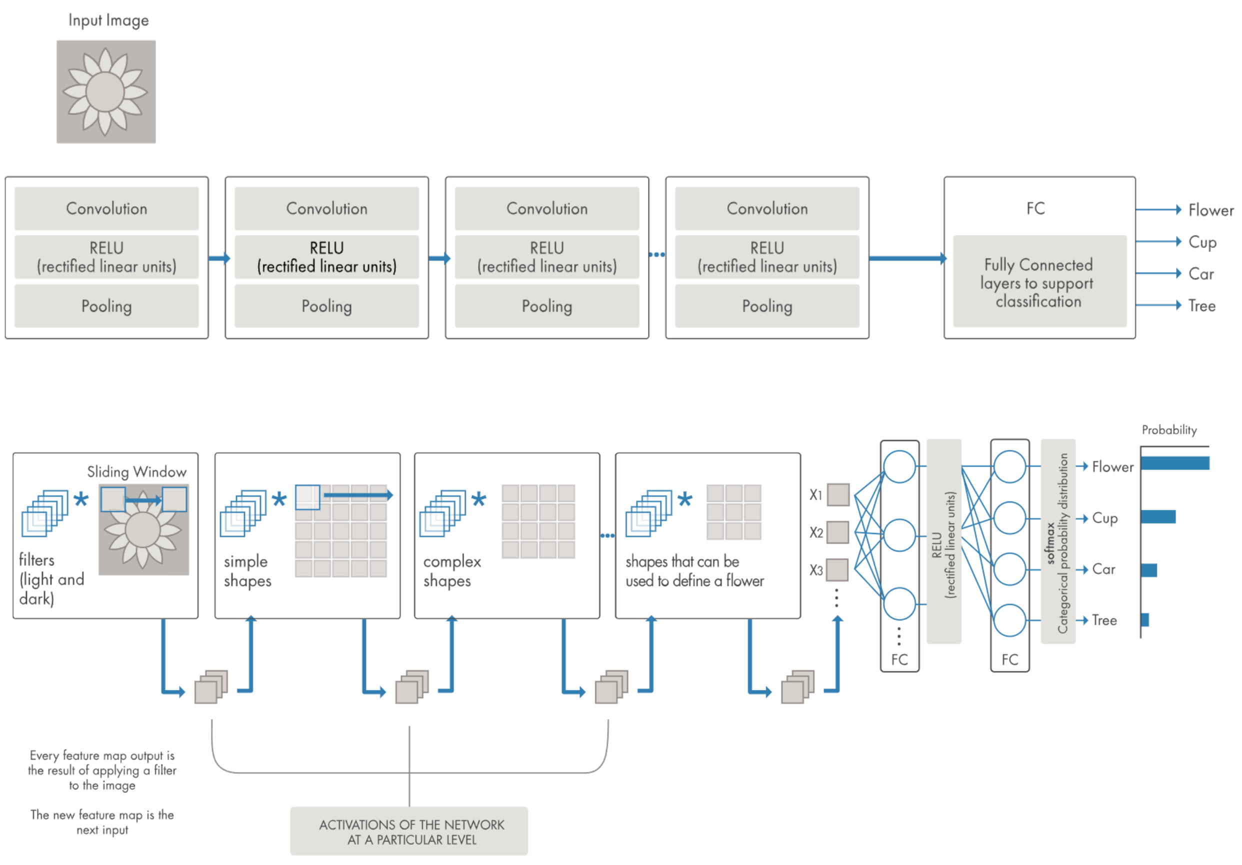

Input Image
Convolution
RELU (rectified linear units)
Pooling
FC
Fully Connected layers to support classification
Flower
Cup
Car
Tree
Sliding Window
filters (light and dark)
simple shapes
complex shapes
shapes that can be used to define a flower
X1
X2
X3
FC
RELU (rectified linear units)
softmax
Categorical probability distribution
Probability
Every feature map output is the result of applying a filter to the image
The new feature map is the next input
ACTIVATIONS OF THE NETWORK AT A PARTICULAR LEVEL

Figure: Example of a network with many convolutional layers. Filters are applied to each training image at different resolutions, and the output of each convolved image serves as the input to the next layer. **(Source -MATLAB)**

What is a Brain tumor?

A brain tumor occurs when abnormal cells form within the brain. There are two main types of tumors: cancerous (malignant) tumors and benign tumors. Cancerous tumors can be divided into primary tumors, which start within the brain, and secondary tumors, which have spread from elsewhere, known as brain metastasis tumors. All types of brain tumors may produce symptoms that vary depending on the part of the brain involved. These symptoms may include headaches, seizures, problems with vision, vomiting and mental changes. The headache is classically worse in the morning and goes away with vomiting. Other symptoms may include difficulty walking, speaking or with sensations. As the disease progresses, unconsciousness may occur.

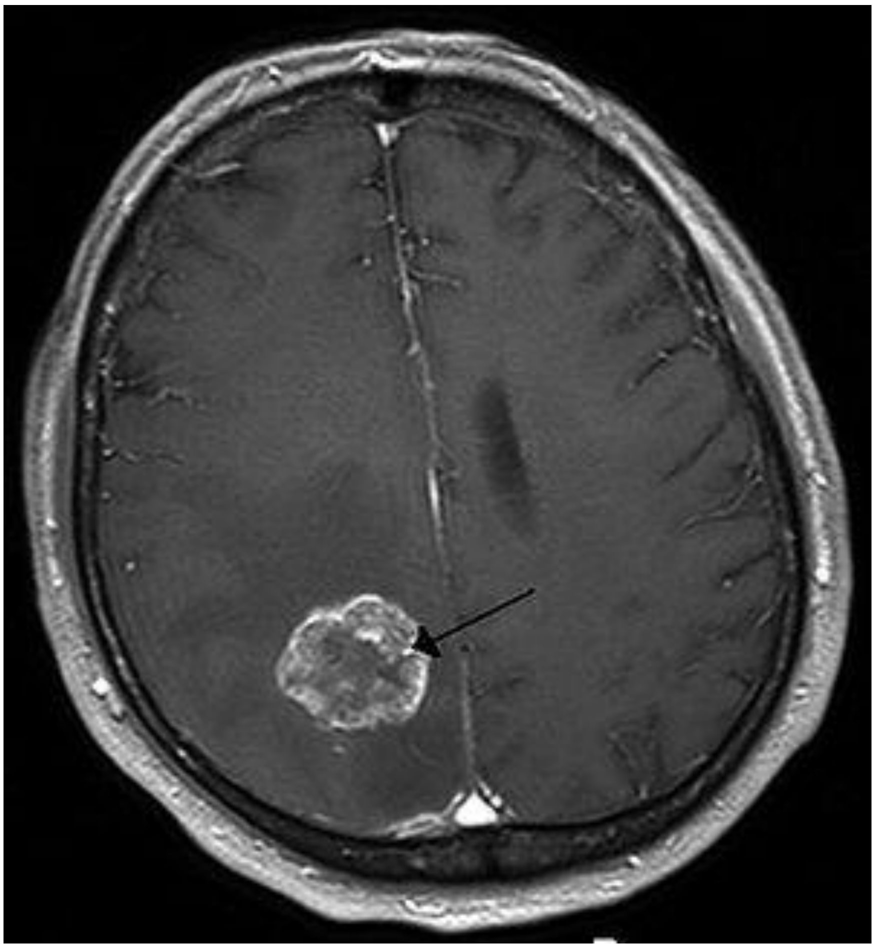


Figure: Brain metastasis in the right cerebral hemisphere, shown on magnetic resonance imaging

Data Set

The image data that is used for this project is Brain MRI Images for Brain Tumor Detection. It is an open-source data available on Kaggle. It consists of 506 high quality images of MRI scans of two classes:

- NO - no tumor, encoded as 0
- YES - tumor, encoded as 1

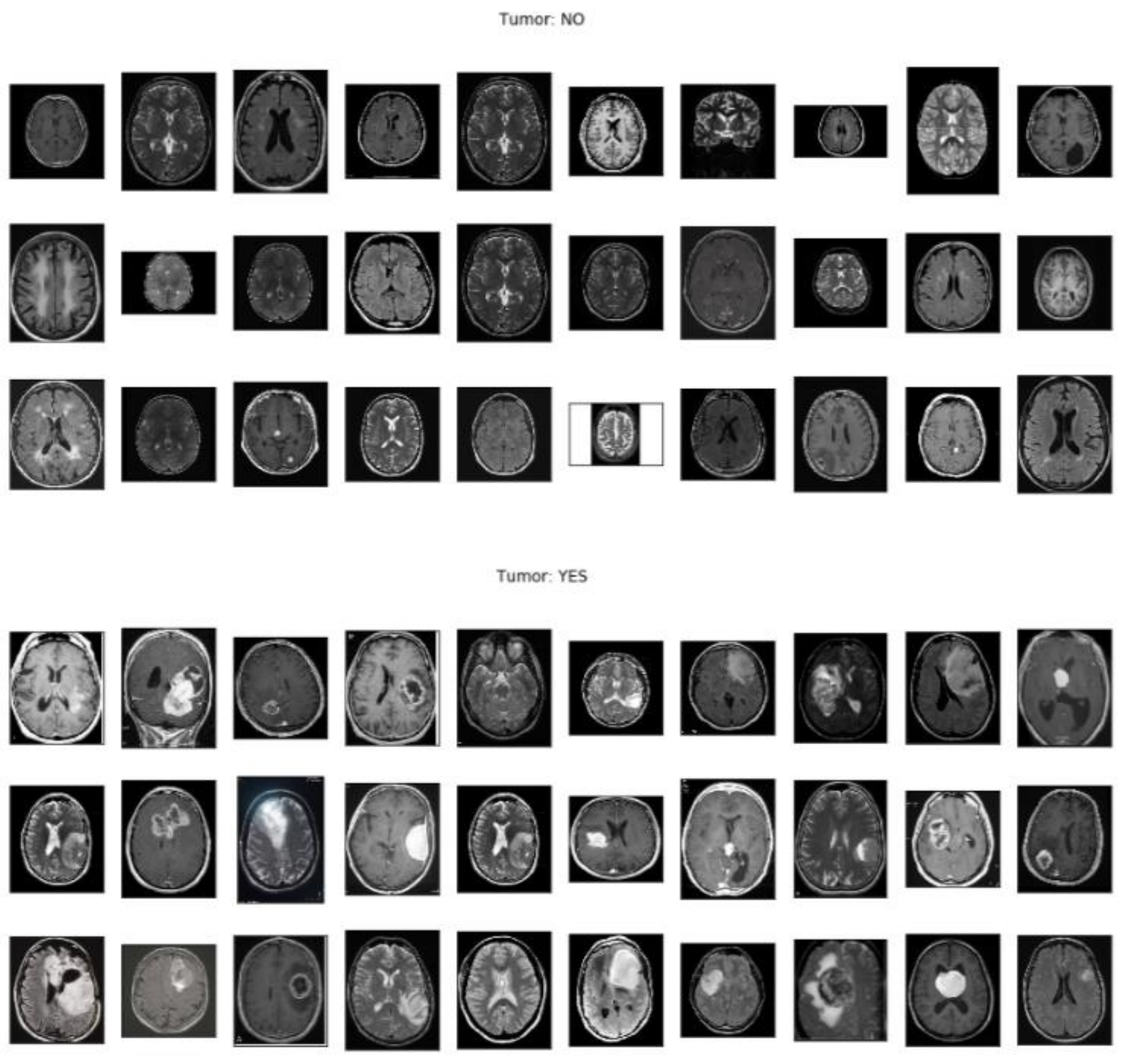
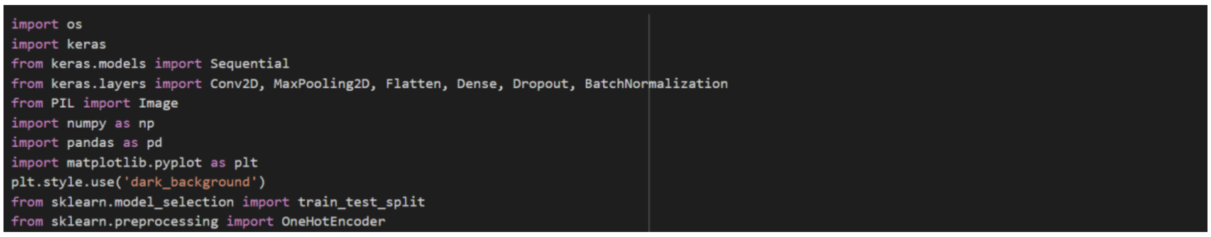


Figure: Brain MRI Images for Brain Tumor Detection DATASET view

Our model will be trained on above data set from Kaggle Brain MRI Images for Brain Tumor Detection. The data consist of 253 images among which 155 images are labeled "yes" and 98 images labeled "no".

Environment setting

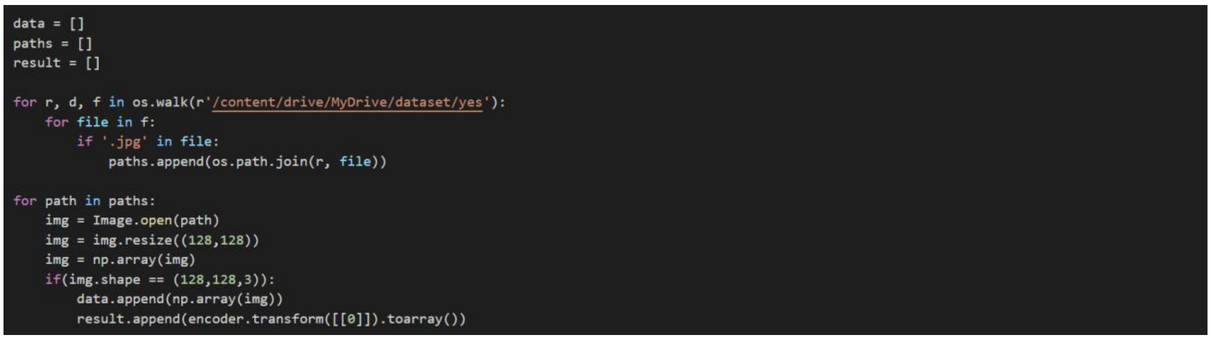
I have used Google Collaboratory for this project. The first step is to setup a suitable environment for training the model. This step includes the importing of different libraries, which will make our work easy apart from that it includes the mounting of the google drive, which includes the labelled MRI image dataset.



```
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
```

Preprocessing the Data

As it is clear from the dataset that there are non-uniform images of various heights and widths. In order to train the model, we have to resize the images to make them compatible for training.



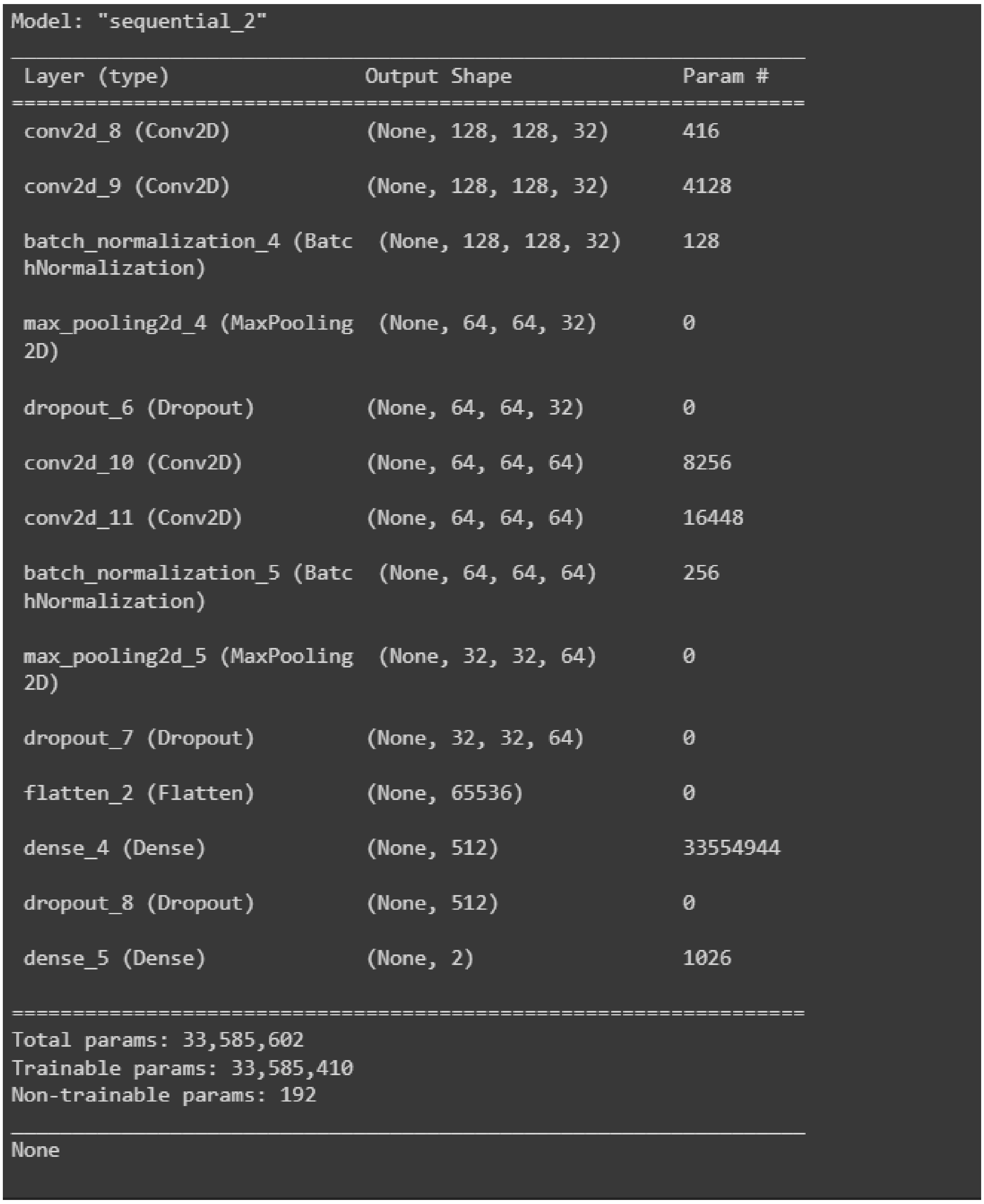
```
data = []
paths = []
result = []

for r, d, f in os.walk(r'/content/drive/MyDrive/dataset/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())
```

Architecture of the CNN model

For this project, we are using a 2D Convolutional Neural Network with 4 layers to classify 2D MRI brain scan images as tumorous or non-tumorous. Our model is Sequential Keras model, which is a plain stack of layers where each layer has exactly one input tensor and one output tensor is composed of standardized 128*128*32 single-channel image arrays. After training, an image passed to the model will produce a prediction of either 1 (tumorous) or 0 (non-tumorous).



```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 128, 128, 32)      416

 conv2d_9 (Conv2D)           (None, 128, 128, 32)      4128

 batch_normalization_4 (Batc  (None, 128, 128, 32)     128
 hNormalization)

 max_pooling2d_4 (MaxPooling  (None, 64, 64, 32)       0
 2D)

 dropout_6 (Dropout)         (None, 64, 64, 32)        0

 conv2d_10 (Conv2D)          (None, 64, 64, 64)        8256

 conv2d_11 (Conv2D)          (None, 64, 64, 64)        16448

 batch_normalization_5 (Batc  (None, 64, 64, 64)       256
 hNormalization)

 max_pooling2d_5 (MaxPooling  (None, 32, 32, 64)       0
 2D)

 dropout_7 (Dropout)         (None, 32, 32, 64)        0

 flatten_2 (Flatten)         (None, 65536)             0

 dense_4 (Dense)             (None, 512)               33554944

 dropout_8 (Dropout)         (None, 512)               0

 dense_5 (Dense)             (None, 2)                 1026

=================================================================
Total params: 33,585,602
Trainable params: 33,585,410
Non-trainable params: 192
_________________________________________________________________
None
```

Summary of the model

- Our model is a **Sequential Keras model**, which is a plain stack of layers where each layer has *exactly one* input tensor and one output tensor.

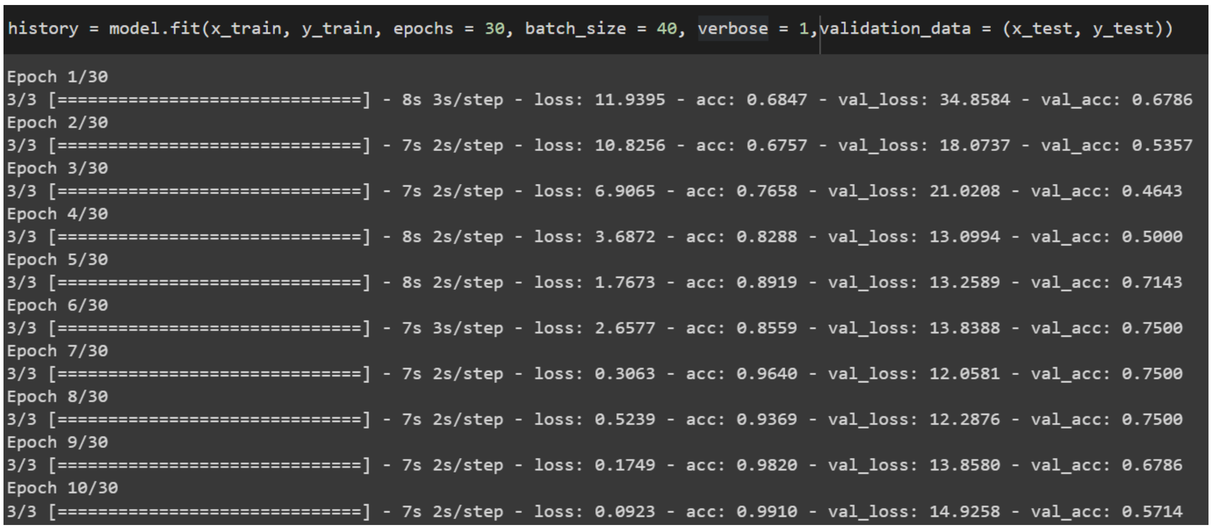
- Within each layer, a 2D convolution layer and a MaxPool 2D is created. The 2D convolution creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. MaxPool **performs pooling** operation for 2D spatial data.

- The activation function used in first 3 layers is **ReLu (Rectified Linear Unit).** We are using ReLU instead of standard sigmoid function because it has better convergence and can train the neural network more accurately, reduces the likelihood of vanishing gradient in back propagation. The ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

- The activation function in the last layer is **SoftMax,** *SoftMax* is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector.

Training

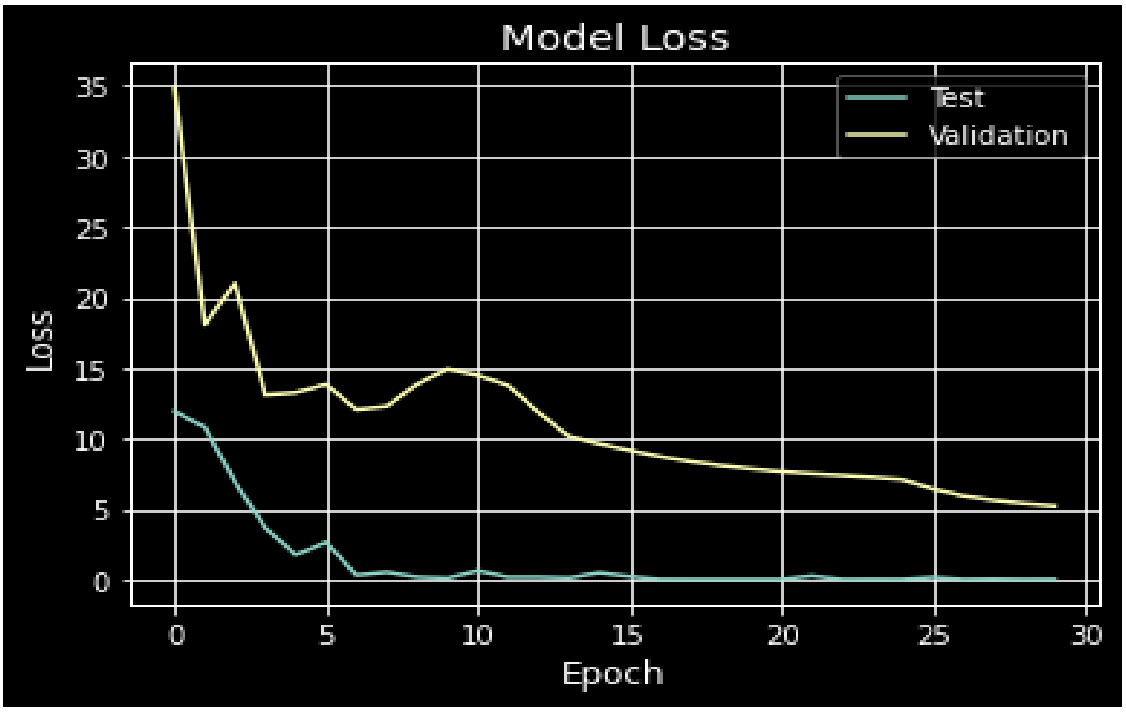
We provide the training data and labels along with validation data and labels, along with batch size and epochs. Batch size is where the training will be done by slicing the data into batch size of 40 and repeatedly iterating over the dataset for a 30 number of epochs.



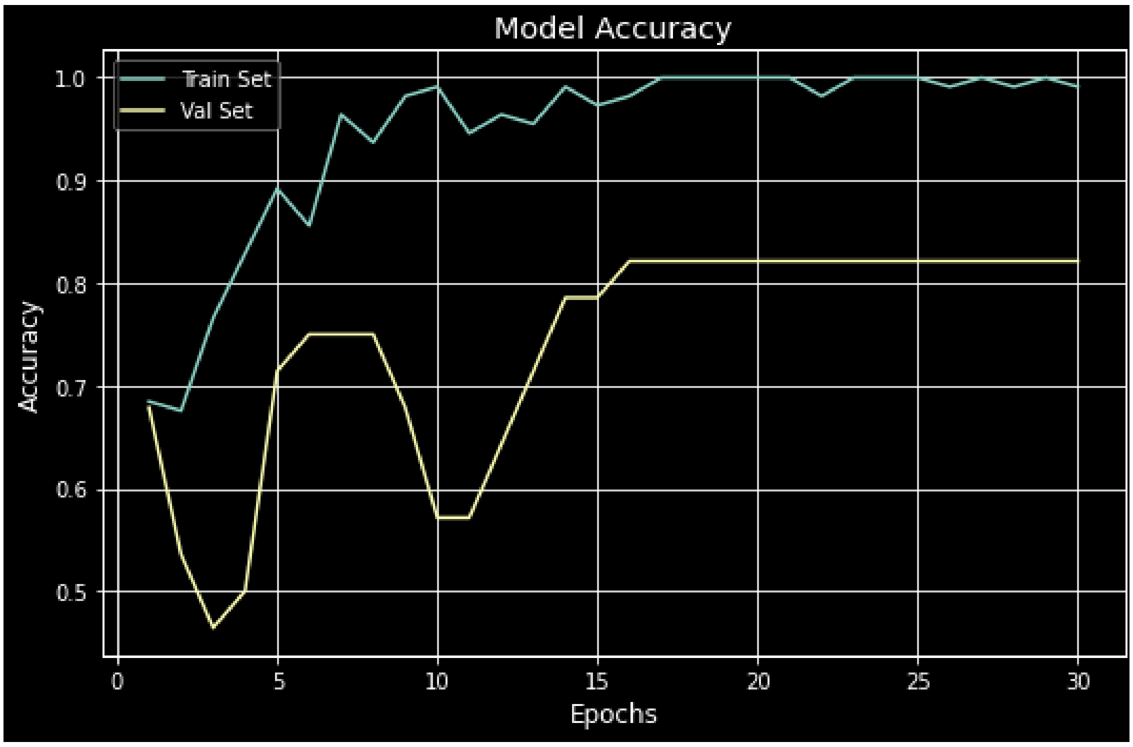
```
history = model.fit(x_train, y_train, epochs = 30, batch_size = 40, verbose = 1,validation_data = (x_test, y_test))

Epoch 1/30
3/3 [==============================] - 8s 3s/step - loss: 11.9395 - acc: 0.6847 - val_loss: 34.8584 - val_acc: 0.6786
Epoch 2/30
3/3 [==============================] - 7s 2s/step - loss: 10.8256 - acc: 0.6757 - val_loss: 18.0737 - val_acc: 0.5357
Epoch 3/30
3/3 [==============================] - 7s 2s/step - loss: 6.9065 - acc: 0.7658 - val_loss: 21.0208 - val_acc: 0.4643
Epoch 4/30
3/3 [==============================] - 8s 2s/step - loss: 3.6872 - acc: 0.8288 - val_loss: 13.0994 - val_acc: 0.5000
Epoch 5/30
3/3 [==============================] - 8s 2s/step - loss: 1.7673 - acc: 0.8919 - val_loss: 13.2589 - val_acc: 0.7143
Epoch 6/30
3/3 [==============================] - 7s 3s/step - loss: 2.6577 - acc: 0.8559 - val_loss: 13.8388 - val_acc: 0.7500
Epoch 7/30
3/3 [==============================] - 7s 2s/step - loss: 0.3063 - acc: 0.9640 - val_loss: 12.0581 - val_acc: 0.7500
Epoch 8/30
3/3 [==============================] - 7s 2s/step - loss: 0.5239 - acc: 0.9369 - val_loss: 12.2876 - val_acc: 0.7500
Epoch 9/30
3/3 [==============================] - 7s 2s/step - loss: 0.1749 - acc: 0.9820 - val_loss: 13.8580 - val_acc: 0.6786
Epoch 10/30
3/3 [==============================] - 7s 2s/step - loss: 0.0923 - acc: 0.9910 - val_loss: 14.9258 - val_acc: 0.5714
```

Result

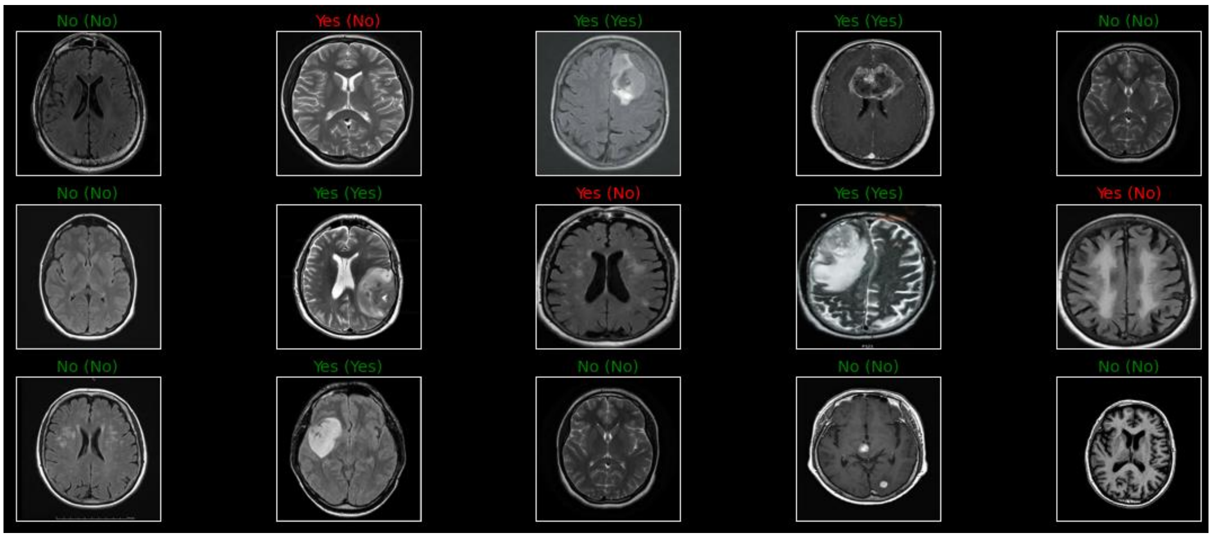


Model Loss
Test
Validation
Loss
Epoch



Model Accuracy
Train Set
Val Set
Accuracy
Epochs

Sneak Peek into Results



Conclusion

Abnormal growth of tissue in the brain which affect the normal functioning of the brain is considered a brain tumor. The main goal of medical image processing is to identify accurate and meaningful information using algorithms with minimum error possible. Brain tumor detection and classification through MRI images is a good application of Deep Learning.

This project was a combination of CNN model classification problem, which is to predict whether the subject has brain tumor or not. The final accuracy is about 82% that is much higher than 50% baseline (random guess). However, it could be increased by larger number of train images or through model hyperparameters tuning.