Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Submitted By:

M V H SAI SRIRAJ	20BAI1224
C MOHAN KRISHNA	20BAI1269

A project report submitted to

Dr. BHARADWAJA KUMAR

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

in partial fulfilment of the requirements for the course of

CSE 1015 – MACHINE LEARNING ESSENTIALS

in

B.Tech. COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE ENGINEERING

APRIL 2022

AKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. Bharadwaja Kumar,** Professor Grade 1, School of Computer Science and Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to**. Dr A. Nayeemulla Khan,** Dean of School of Computer Science VIT Chennai, for extending the facilities of the school towards our project and for their unstinting support.

We express our thanks to our Head of the Department **Dr. Priyadarshini** for her support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

M V H SAI SRIRAJ
(20BAI1224)

C MOHAN KRISHNA
(20BAI1269)

ABSTRACT

A tumor is a solid mass of tissue that forms when abnormal cells group together. A cancerous or non-cancerous mass or growth of abnormal cells in the brain. Tumors can start in the brain, or cancer elsewhere in the body can spread to the brain.

Many tumors are not cancer . A brain tumor is a cluster of abnormal cells that grows out of control in your brain. Some brain tumors are benign, which means the cells aren't cancer. Brain tumors are called primary tumors if they started in your brain. They're considered secondary if they started somewhere else in your body and spread to your brain.

Symptoms of brain tumors vary according to the type of tumor and the location. Because different areas of the brain control different functions of the body, where the tumor lies affects the symptoms you get. So you cannot predict the tumour type by examine the type of symptoms you are getting

This project aims to use the MRI Images of the brain and classifies it as to three main tumor types i.e Meningioma, Glioma and Pituitary Tumor. The Classification of the MRI Images helps the patient to proceed with a pre doctor aid as well as the doctor to go with the treatment of the patient without any further confusion.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION AND BACKGROUND STUDY

A tumor is a solid mass of tissue that forms when abnormal cells group together. A cancerous or non-cancerous mass or growth of abnormal cells in the brain. Tumors can start in the brain, or cancer elsewhere in the body can spread to the brain.

Many tumors are not cancer . A brain tumor is a cluster of abnormal cells that grows out of control in your brain. Some brain tumors are benign, which means the cells aren’t cancer. Brain tumors are called primary tumors if they started in your brain. They’re considered secondary if they started somewhere else in your body and spread to your brain.

Symptoms of brain tumors vary according to the type of tumor and the location. Because different areas of the brain control different functions of the body, where the tumor lies affects the symptoms you get.

- Headaches, which may not get better with the standard migraine cures. You might see you're getting them on a more regular basis or they're more regrettable than expected
- Seizures, particularly in a person who doesn't have a history of seizures
- Changes in speech or hearing
- Changes in vision
- Balance problems
- Problems with walking

Background study

MENINGIOMA :

Meningioma is the most common type of primary brain tumor, accounting for approximately 30 percent of all brain tumors.

- About 85% of meningiomas are noncancerous, slow-growing tumors
- Meningioma originates from the meningeal layers of either the brain or the spinal cord
- Machine learning analyses can differentiate meningioma grade by features on magnetic resonance imaging
- While magnetic resonance imaging (MRI), is the standard radiologic technique for provisional diagnosis and surveillance of meningioma

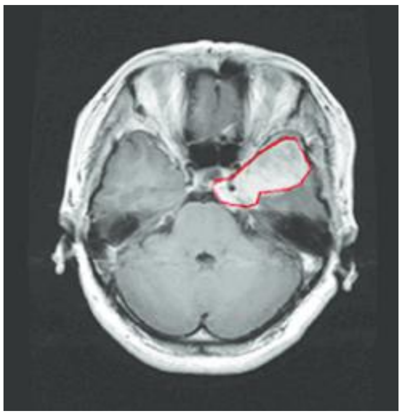
GLIOMA :

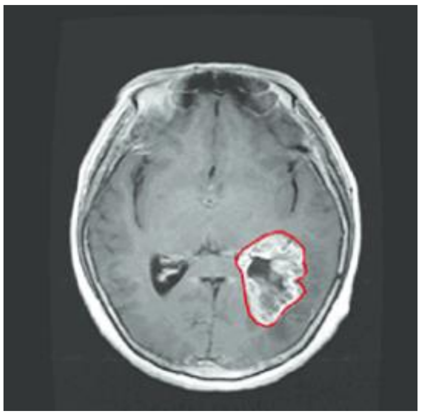
- Gliomas are brain tumours that start in glial cells. These are the supporting cells of the brain and the spinal cord
- Machine learning has been applied to the analysis of MRI data in glioma research and has the potential to change clinical practice and improve patient outcomes.

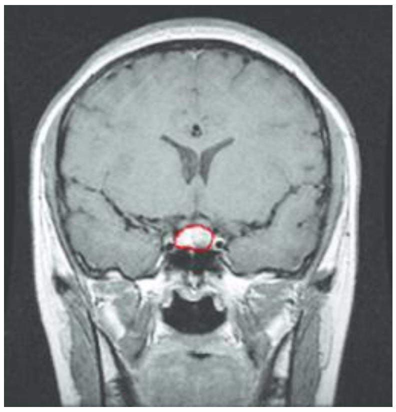
- This systematic review synthesizes and analyzes the current state of machine learning applications to glioma MRI data and explores the use of machine learning for systematic review automation
- To truly profit from the power of machine learning in glioma imaging, larger amounts of well-annotated data are required to achieve good generalization across datasets. Data sharing in medical research is being actively pursued in an effort to raise the integrity and accelerate the pace of research
- MR images and histopathology are the most important indicators in the determination of whether it is gliomas or other abnormalities. Machine learning methods have good performance at pattern recognition of images.

PITUITARY TUMOR

- Pituitary tumors are abnormal growths that develop in your pituitary gland. Some pituitary tumors result in too much of the hormones that regulate important functions of your body
- Pituitary tumors are abnormal growths that develop in your pituitary gland. Some pituitary tumors result in too much of the hormones that regulate important functions of your body
- Machine learning (ML) have garnered increasing attention to quantitatively analyze complex medical data to improve individualized care for patients with Pituitary tumors
- ML can be used to design andtrain software algorithms to learn from and act on data.
- The type of PA cannot be clearly recognized by preoperative MRIbut can be classified by immunohistochemical staining of resected tumor samples after surgery
- MR imaging could potentially serve as a novel approach to preoperatively predict visual recovery and allow personalized counseling for individual pituitary adenoma patients.
- Machine learning algorithms was proposed with the feasible predictive performance for postoperative visual recovery, allowing personalized counseling for pituitary adenoma patients.


(Meningioma Tumor)


(Glioma Tumor)


(Pituitary Tumor)

CHAPTER 2: PROPOSED METHODOLOGY

Our Project aims to do the comparison study between the three famous Image Classification Algorithms that is VGG 16 , Resnet 50 and Efficient Net B series Algorithms.

This project aims to use the MRI Images of the brain and classifies it as to three main tumor types i.e Meningioma, Glioma and Pituitary Tumor. The Classification of the MRI Images helps the patient to proceed with a pre doctor aid as well as the doctor to go with the treatment of the patient without any further confusion.

VGG Algorithm :

Visual Geometry Group is an acronym for Visual Geometry Group (a group of researchers at Oxford who developed this architecture). The VGG architecture is divided into blocks, each of which is made up of 2D Convolution and Max Pooling layers. VGGNet is available in two varieties: VGG16 and VGG19, which differ in the number of layers they contain.

The ability of a Convolutional Neural Network (CNN) to fit more complex functions increases as the number of layers increases. As a result, having more layers is always preferable (not to be confused with an artificial neural network which does not necessarily give a significantly better performance with increase in number of hidden layers).

The backpropagation algorithm is used to update the weights of a neural network. The backpropagation algorithm makes a tiny adjustment to each weight in order to reduce the model's loss. What causes this to happen? It refreshes each weight in such a way that the loss progresses in the right direction. This direction is nothing more than the weight's gradient (with respect to the loss).

We can find this gradient for each weight using the chain rule. It's the same as (local gradient) x. (gradient flowing from ahead)

This is where the issue arises. This value keeps getting multiplied by each local gradient as this gradient flows backward to the initial layers. As a result, the gradient gets smaller and smaller, making the updates more difficult.

VGG 16 was proposed by Karen Simonyan and Andrew Zisserman of the Visual Geometry Group Lab of Oxford University in 2014 in the paper “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION”. CNN is a deep learning algorithm which takes input image, assigns weights according to the features in the image and be able to differentiate one from another. CNN is used for image recognition, object classification and face recognition.

Resnet Algorithm :

ResNet-50 is a residual network. A residual network is a type of DAG network that has residual (or shortcut) connections that bypass the main network layers.

Residual connections enable the parameter gradients to propagate more easily from the output layer to the earlier layers of the network, which makes it possible to train deeper networks. This increased network depth can result in higher accuracies on more difficult tasks.

ResNet50 is a variant of ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. It has 3.8 x 10^9 Floating points operations.

ResNet50 is a variant of ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. It has 3.8 x 10^9 Floating points operations

The ResNets were initially applied to the image recognition task but as it is mentioned in the paper that the framework can also be used for non computer vision tasks also to achieve better accuracy.

This architecture can be used on computer vision tasks such as image classification, object localisation, object detection and this framework can also be applied to non computer vision tasks to give them the benefit of depth and to reduce the computational expense also.

The ResNet architecture (seen below) should now make perfect sense in terms of how it prevents the vanishing gradient problem. ResNet is an abbreviation for Residual Network.

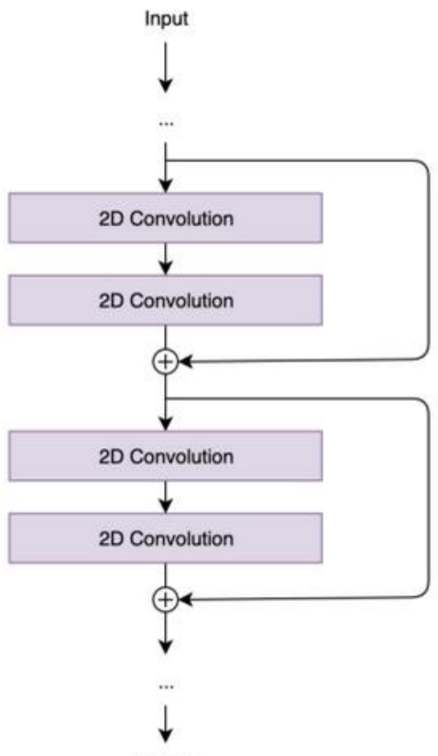
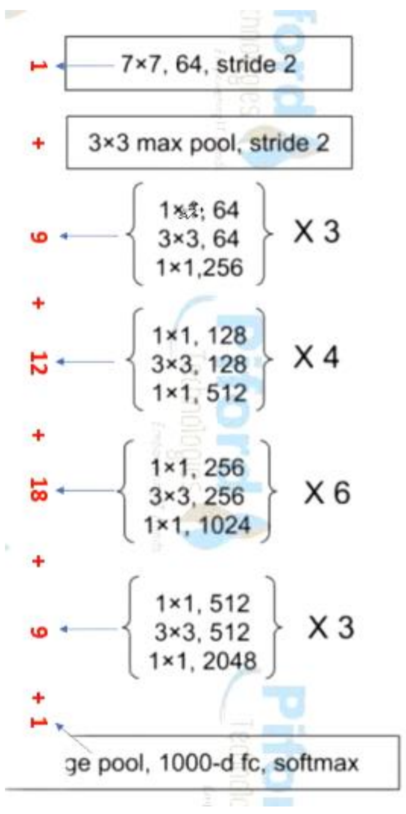
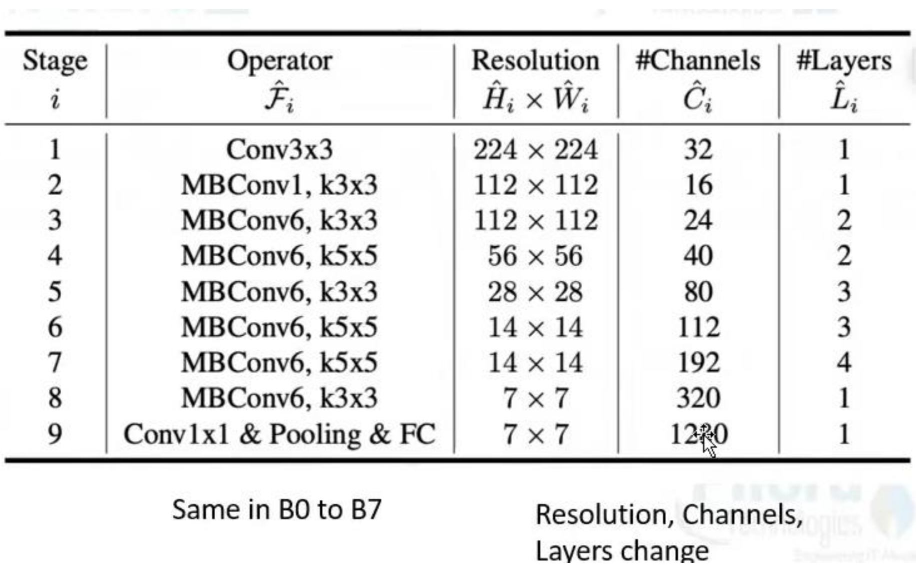


Fig. 4 ResNet architecture



Efficient Net B series Algorithm :

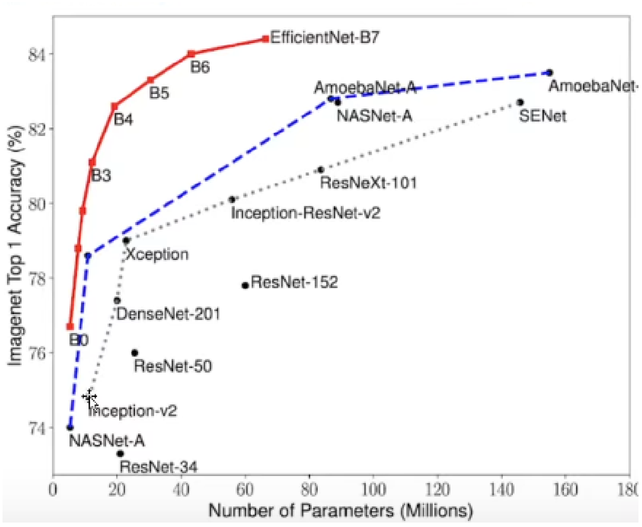
All EfficientNet models are scaled from our baseline EfficientNet-B0 using different compound coefficients, resulting in EfficientNet performance results on ImageNet. The EfficientNet-B0 architecture was created by the neural network itself, not by engineers. They used a multi-objective neural architecture search to create this model, which maximises both accuracy and floating-point operations. Accepting B0 as a benchmark model, the developers developed a series of EfficientNets ranging from B1 to B7 that achieved cutting-edge precision on ImageNet while outperforming its competition. EfficientB0 has 230 layers and 7 MBConv blocks, and it has a thick block structure with four tightly linked layers and a development rate of four.



Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Same in B0 to B7

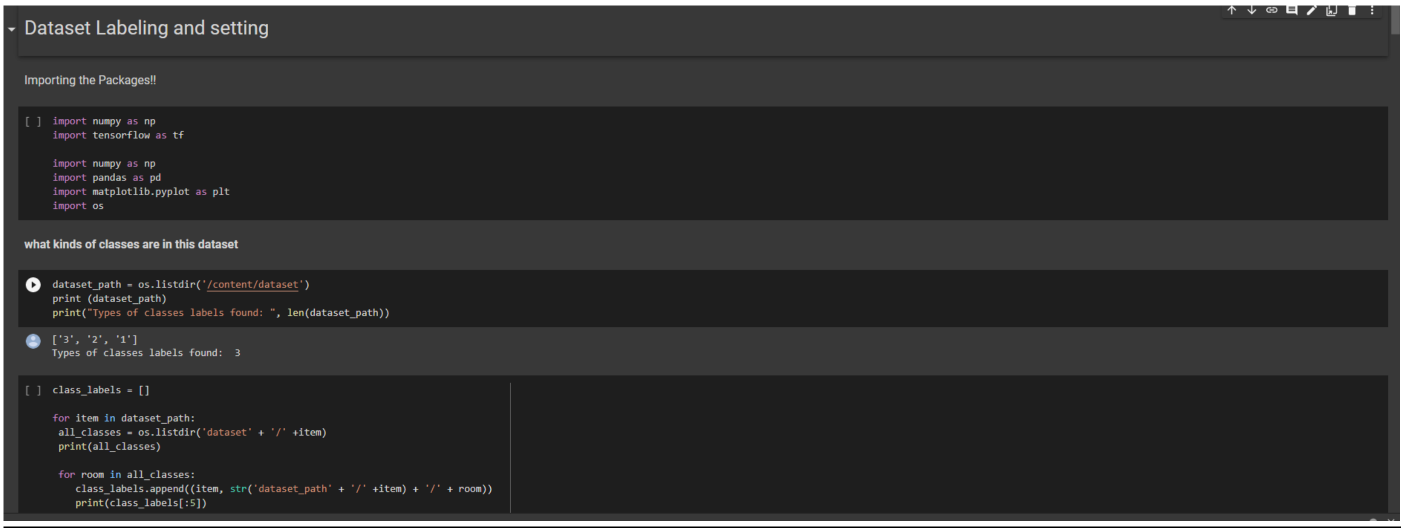
Resolution, Channels, Layers change



CHAPTER 3: IMPLEMENTATION OF THE MODEL

Link of the ipynb notebook:

https://colab.research.google.com/drive/1VvcXMfhhtDgO9SF8T327h8dZzfkIs6w8?usp=sharing



Dataset Labeling and setting

Importing the Packages!!

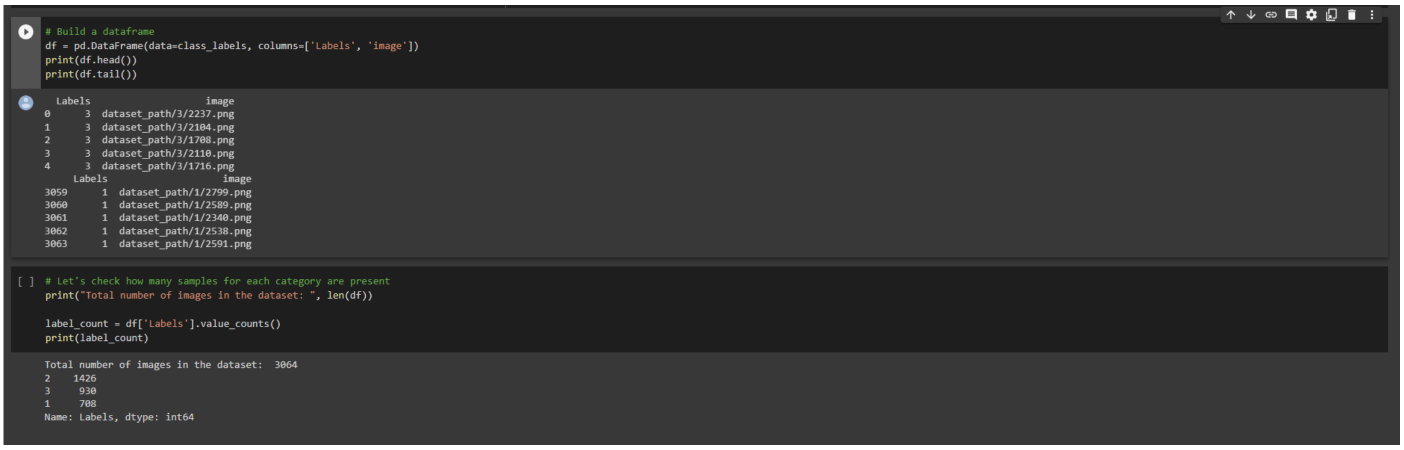
```
import numpy as np
import tensorflow as tf

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
```

what kinds of classes are in this dataset

```
dataset_path = os.listdir('/content/dataset')
print (dataset_path)
print("Types of classes labels found: ", len(dataset_path))
```

```
['3', '2', '1']
Types of classes labels found:  3
```

```
class_labels = []

for item in dataset_path:
 all_classes = os.listdir('dataset' + '/' +item)
 print(all_classes)

 for room in all_classes:
    class_labels.append((item, str('dataset_path' + '/' +item) + '/' + room))
    print(class_labels[:5])
```

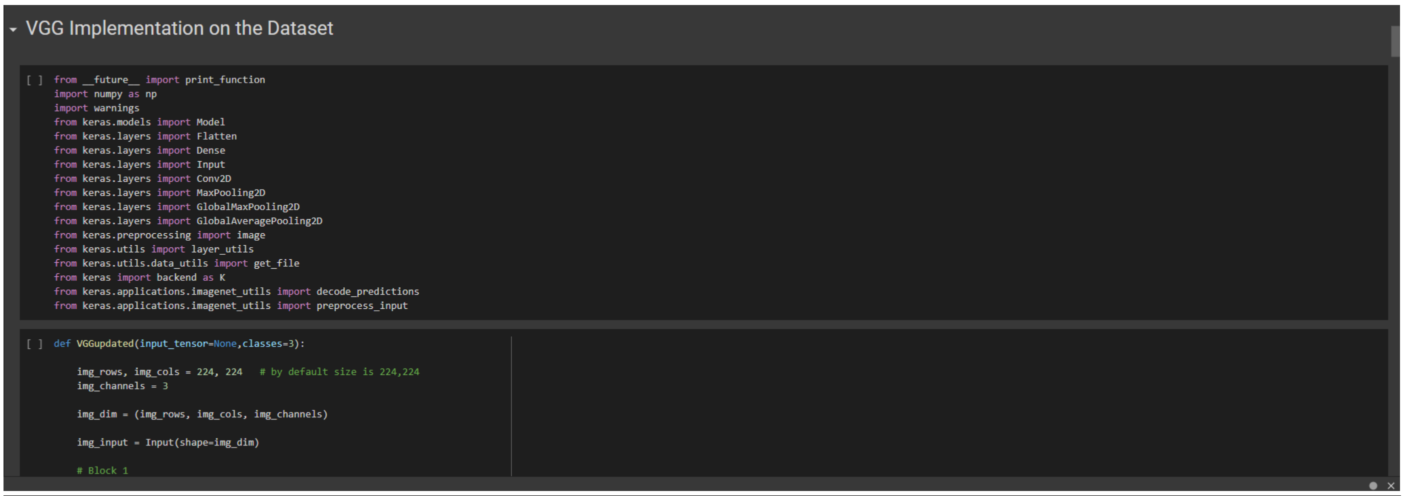


```
# Build a dataframe
df = pd.DataFrame(data=class_labels, columns=['Labels', 'image'])
print(df.head())
print(df.tail())
```

```
  Labels                   image
0      3  dataset_path/3/2237.png
1      3  dataset_path/3/2104.png
2      3  dataset_path/3/1708.png
3      3  dataset_path/3/2110.png
4      3  dataset_path/3/1716.png
     Labels                   image
3059      1  dataset_path/1/2799.png
3060      1  dataset_path/1/2589.png
3061      1  dataset_path/1/2340.png
3062      1  dataset_path/1/2538.png
3063      1  dataset_path/1/2591.png
```

```
# Let's check how many samples for each category are present
print("Total number of images in the dataset: ", len(df))

label_count = df['Labels'].value_counts()
print(label_count)
```

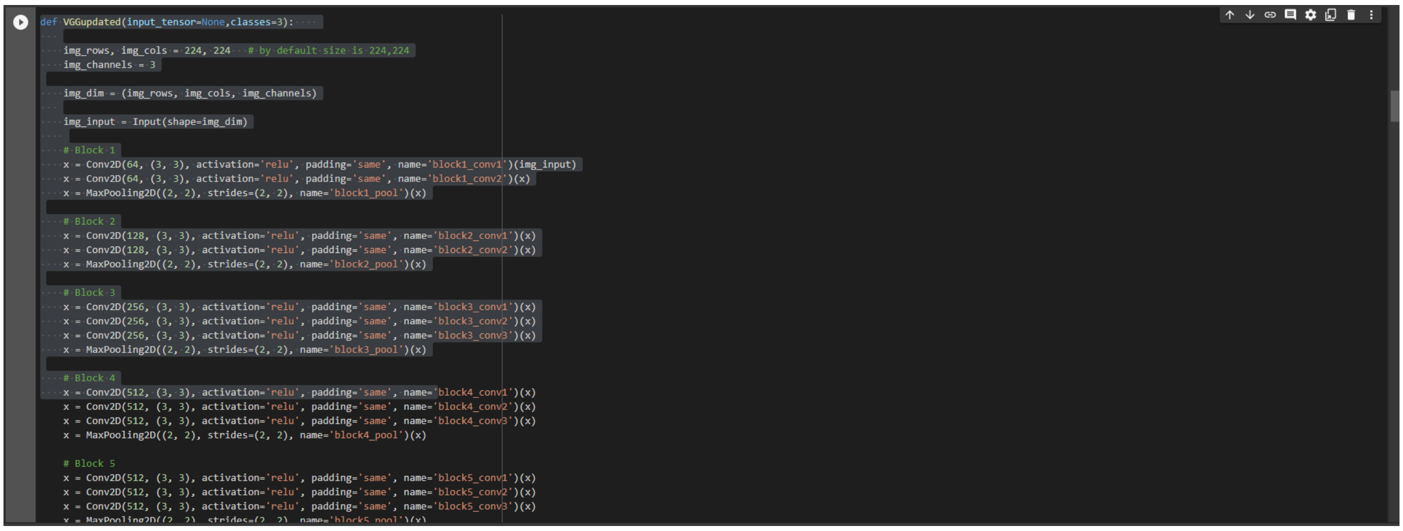
```
Total number of images in the dataset:  3064
2    1426
3     930
1     708
Name: Labels, dtype: int64
```

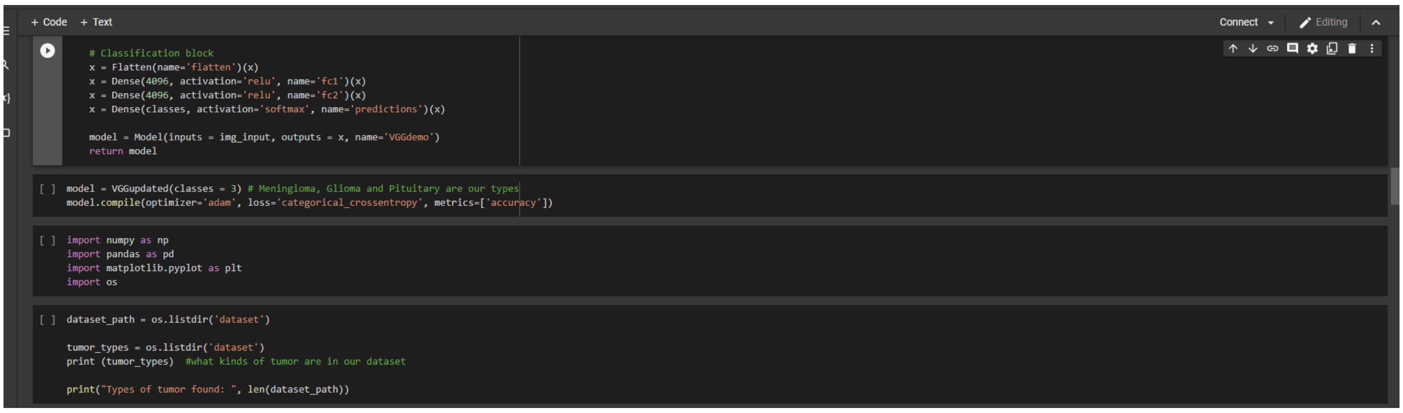


VGG Implementation on the Dataset

```
from __future__ import print_function
import numpy as np
import warnings
from keras.models import Model
from keras.layers import Flatten
from keras.layers import Dense
from keras.layers import Input
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import GlobalMaxPooling2D
from keras.layers import GlobalAveragePooling2D
from keras.preprocessing import image
from keras.utils import layer_utils
from keras.utils.data_utils import get_file
from keras import backend as K
from keras.applications.imagenet_utils import decode_predictions
from keras.applications.imagenet_utils import preprocess_input
```

```
def VGGupdated(input_tensor=None,classes=3):

    img_rows, img_cols = 224, 224   # by default size is 224,224
    img_channels = 3

    img_dim = (img_rows, img_cols, img_channels)

    img_input = Input(shape=img_dim)

    # Block 1
```


```python
def VGGupdated(input_tensor=None,classes=3):

    img_rows, img_cols = 224, 224   # by default size is 224,224
    img_channels = 3

    img_dim = (img_rows, img_cols, img_channels)

    img_input = Input(shape=img_dim)

    # Block 1
    x = Conv2D(64, (3, 3), activation='relu', padding='same', name='block1_conv1')(img_input)
    x = Conv2D(64, (3, 3), activation='relu', padding='same', name='block1_conv2')(x)
    x = MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool')(x)

    # Block 2
    x = Conv2D(128, (3, 3), activation='relu', padding='same', name='block2_conv1')(x)
    x = Conv2D(128, (3, 3), activation='relu', padding='same', name='block2_conv2')(x)
    x = MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool')(x)

    # Block 3
    x = Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv1')(x)
    x = Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv2')(x)
    x = Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv3')(x)
    x = MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool')(x)

    # Block 4
    x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv1')(x)
    x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv2')(x)
    x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv3')(x)
    x = MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool')(x)

    # Block 5
    x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv1')(x)
    x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv2')(x)
    x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv3')(x)
    x = MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool')(x)
```

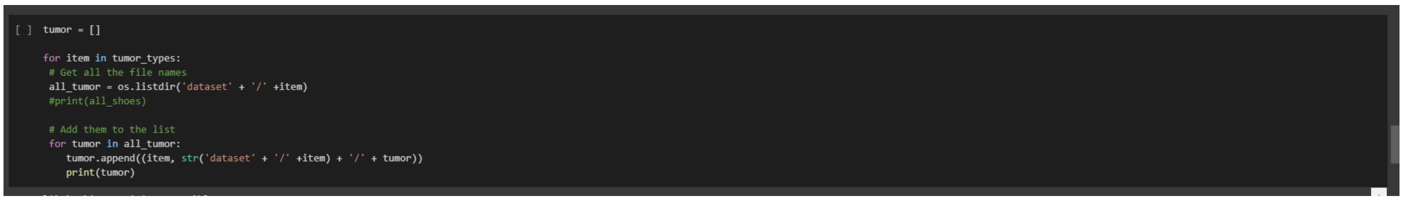


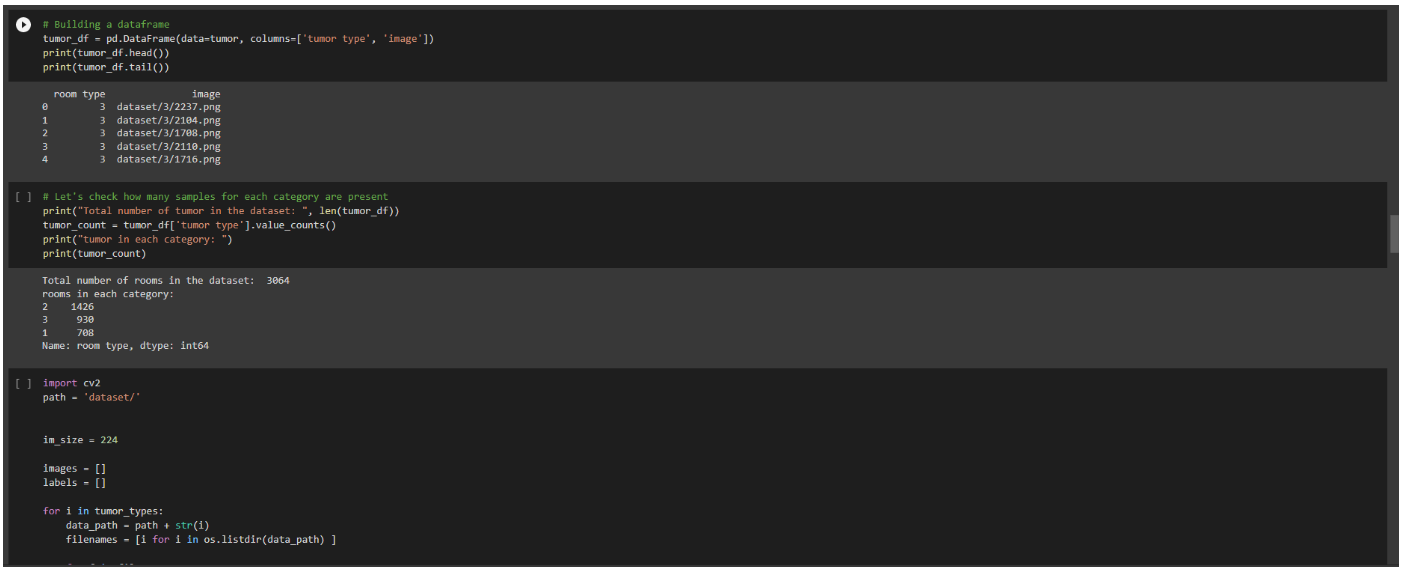
```python
    # Classification block
    x = Flatten(name='flatten')(x)
    x = Dense(4096, activation='relu', name='fc1')(x)
    x = Dense(4096, activation='relu', name='fc2')(x)
    x = Dense(classes, activation='softmax', name='predictions')(x)

    model = Model(inputs = img_input, outputs = x, name='VGGdemo')
    return model
```

```python
model = VGGupdated(classes = 3) # Meningioma, Glioma and Pituitary are our types
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
```

```python
dataset_path = os.listdir('dataset')

tumor_types = os.listdir('dataset')
print (tumor_types)  #what kinds of tumor are in our dataset

print("Types of tumor found: ", len(dataset_path))
```


```python
tumor = []

for item in tumor_types:
 # Get all the file names
 all_tumor = os.listdir('dataset' + '/' +item)
 #print(all_shoes)

 # Add them to the list
 for tumor in all_tumor:
    tumor.append((item, str('dataset' + '/' +item) + '/' + tumor))
    print(tumor)
```

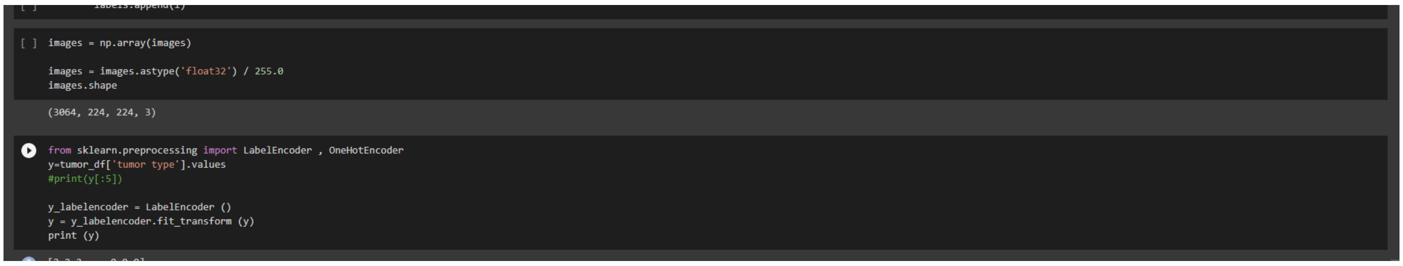

```python
# Building a dataframe
tumor_df = pd.DataFrame(data=tumor, columns=['tumor type', 'image'])
print(tumor_df.head())
print(tumor_df.tail())
```

```
  room type                 image
0         3  dataset/3/2237.png
1         3  dataset/3/2104.png
2         3  dataset/3/1708.png
3         3  dataset/3/2110.png
4         3  dataset/3/1716.png
```

```python
# Let's check how many samples for each category are present
print("Total number of tumor in the dataset: ", len(tumor_df))
tumor_count = tumor_df['tumor type'].value_counts()
print("tumor in each category: ")
print(tumor_count)
```

```
Total number of rooms in the dataset:  3064
rooms in each category:
2    1426
3     930
1     708
Name: room type, dtype: int64
```

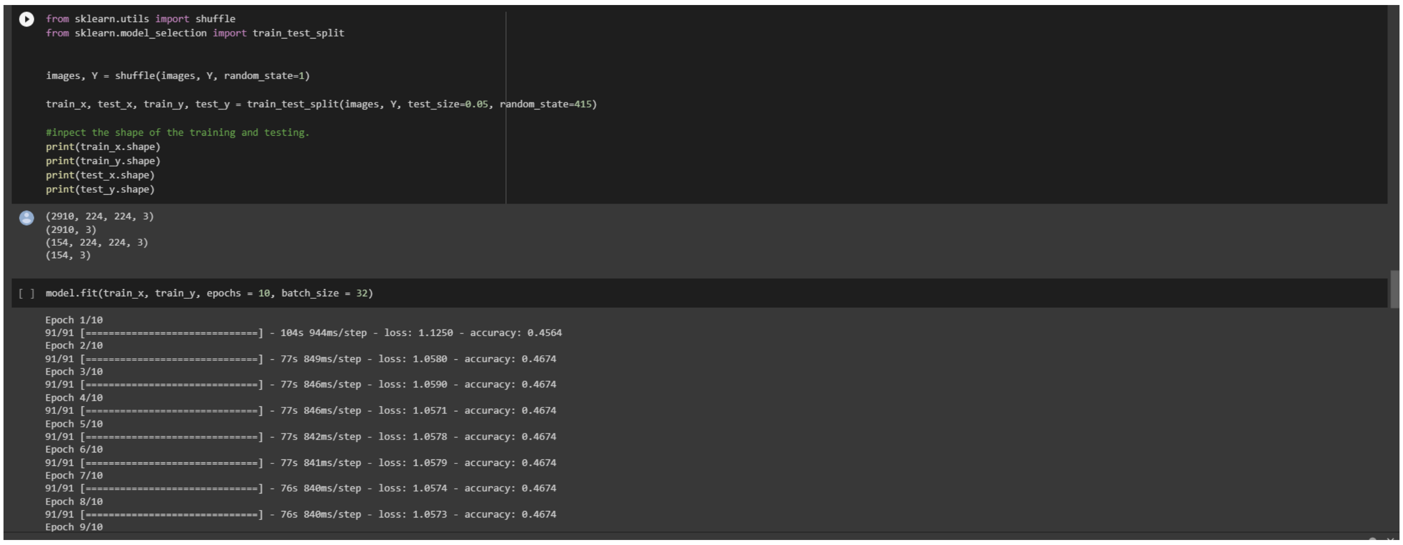
```python
import cv2
path = 'dataset/'


im_size = 224

images = []
labels = []

for i in tumor_types:
    data_path = path + str(i)
    filenames = [i for i in os.listdir(data_path) ]
```


```python
        labels.append(i)
```

```python
images = np.array(images)

images = images.astype('float32') / 255.0
images.shape
```

```
(3064, 224, 224, 3)
```

```python
from sklearn.preprocessing import LabelEncoder , OneHotEncoder
y=tumor_df['tumor type'].values
#print(y[:5])

y_labelencoder = LabelEncoder ()
y = y_labelencoder.fit_transform (y)
print (y)
```



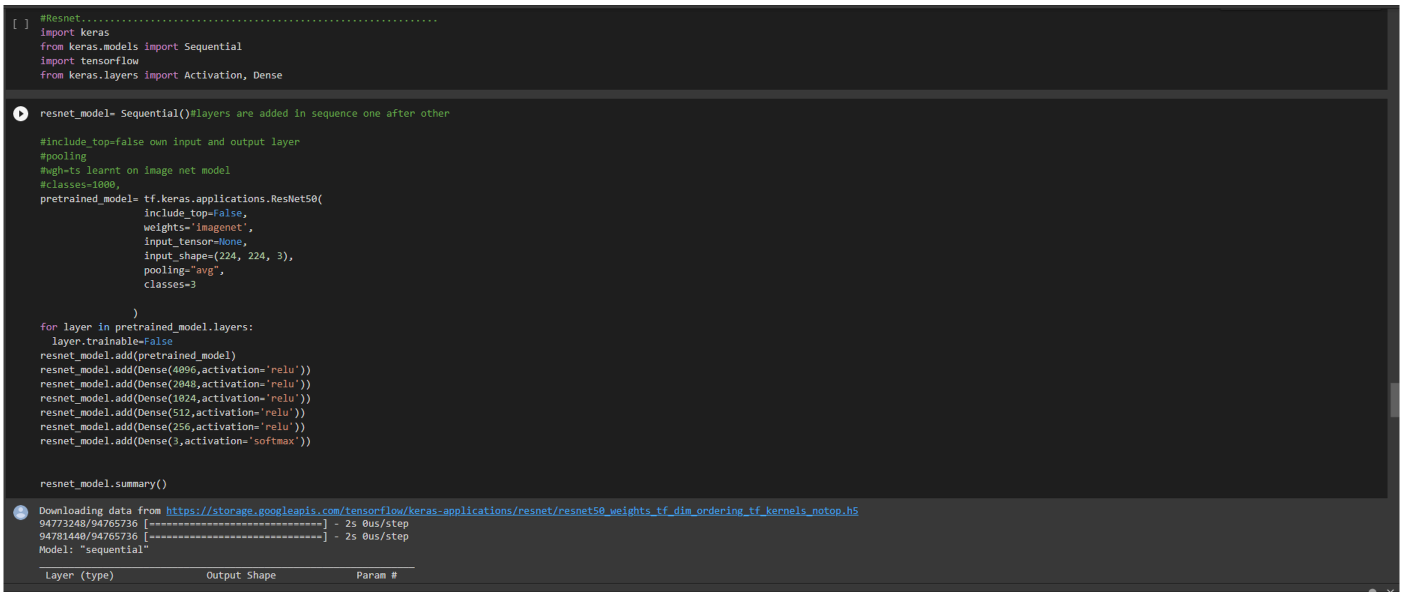
```
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split


images, Y = shuffle(images, Y, random_state=1)

train_x, test_x, train_y, test_y = train_test_split(images, Y, test_size=0.05, random_state=415)

#inpect the shape of the training and testing.
print(train_x.shape)
print(train_y.shape)
print(test_x.shape)
print(test_y.shape)
```

```
(2910, 224, 224, 3)
(2910, 3)
(154, 224, 224, 3)
(154, 3)
```

```
[ ] model.fit(train_x, train_y, epochs = 10, batch_size = 32)
```

```
Epoch 1/10
91/91 [==============================] - 104s 944ms/step - loss: 1.1250 - accuracy: 0.4564
Epoch 2/10
91/91 [==============================] - 77s 849ms/step - loss: 1.0580 - accuracy: 0.4674
Epoch 3/10
91/91 [==============================] - 77s 846ms/step - loss: 1.0590 - accuracy: 0.4674
Epoch 4/10
91/91 [==============================] - 77s 846ms/step - loss: 1.0571 - accuracy: 0.4674
Epoch 5/10
91/91 [==============================] - 77s 842ms/step - loss: 1.0578 - accuracy: 0.4674
Epoch 6/10
91/91 [==============================] - 77s 841ms/step - loss: 1.0579 - accuracy: 0.4674
Epoch 7/10
91/91 [==============================] - 76s 840ms/step - loss: 1.0574 - accuracy: 0.4674
Epoch 8/10
91/91 [==============================] - 76s 840ms/step - loss: 1.0573 - accuracy: 0.4674
Epoch 9/10
```

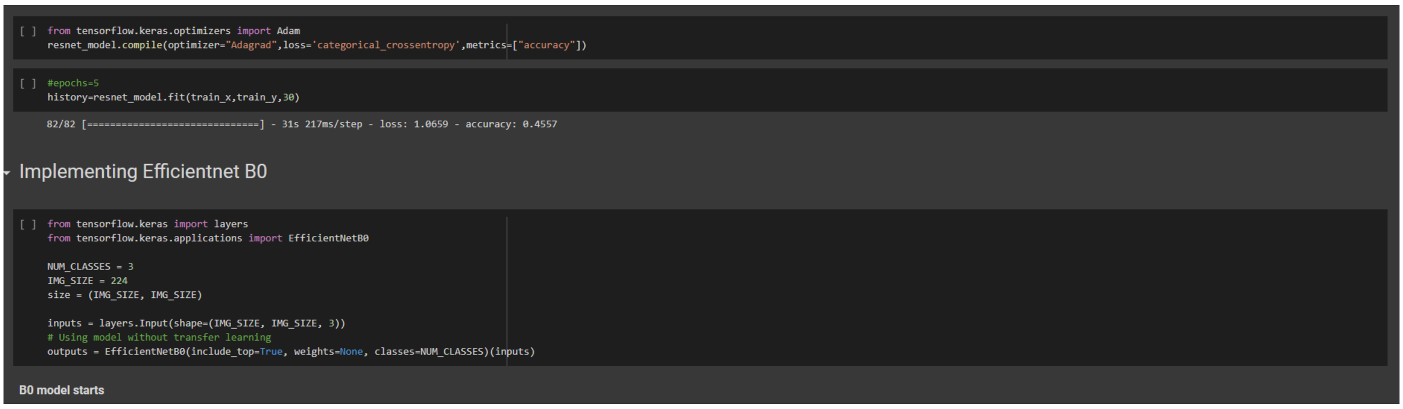
//



```
[ ] #Resnet.............................................................
    import keras
    from keras.models import Sequential
    import tensorflow
    from keras.layers import Activation, Dense
```

```
resnet_model= Sequential()#layers are added in sequence one after other

#include_top=false own input and output layer
#pooling
#wgh=ts learnt on image net model
#classes=1000,
pretrained_model= tf.keras.applications.ResNet50(
                  include_top=False,
                  weights='imagenet',
                  input_tensor=None,
                  input_shape=(224, 224, 3),
                  pooling="avg",
                  classes=3

                )
for layer in pretrained_model.layers:
  layer.trainable=False
resnet_model.add(pretrained_model)
resnet_model.add(Dense(4096,activation='relu'))
resnet_model.add(Dense(2048,activation='relu'))
resnet_model.add(Dense(1024,activation='relu'))
resnet_model.add(Dense(512,activation='relu'))
resnet_model.add(Dense(256,activation='relu'))
resnet_model.add(Dense(3,activation='softmax'))


resnet_model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94773248/94765736 [==============================] - 2s 0us/step
94781440/94765736 [==============================] - 2s 0us/step
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
```



```
[ ] from tensorflow.keras.optimizers import Adam
    resnet_model.compile(optimizer="Adagrad",loss='categorical_crossentropy',metrics=["accuracy"])
```

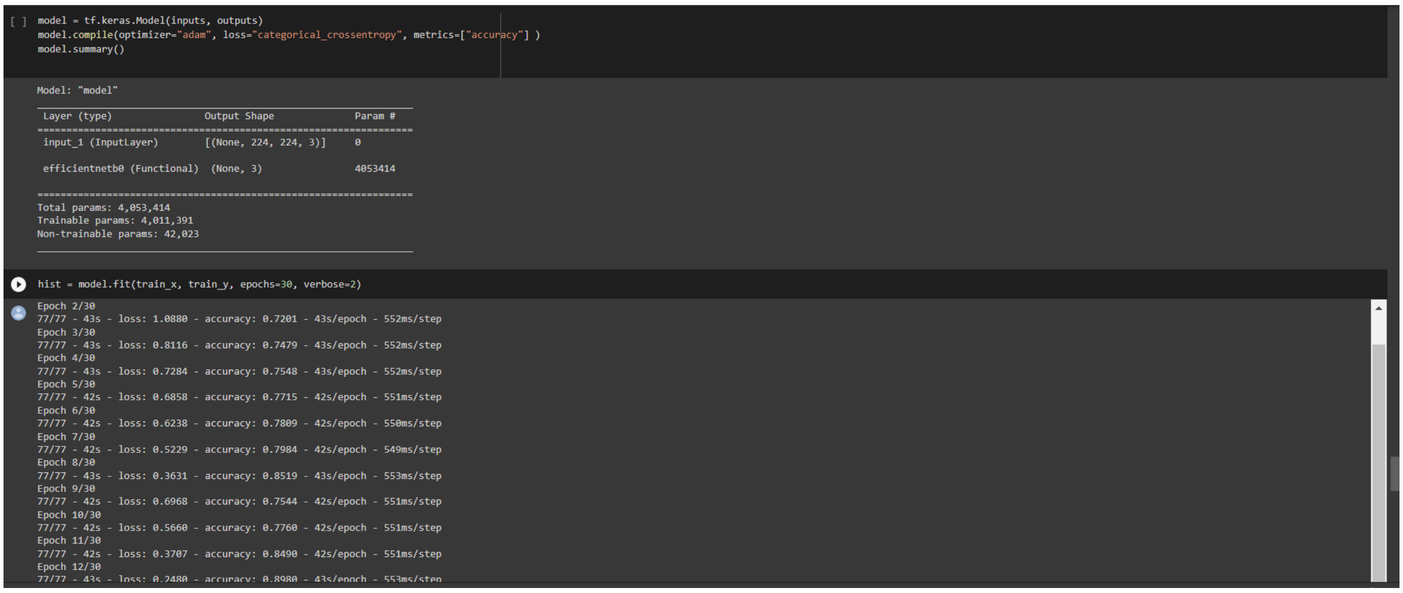
```
[ ] #epochs=5
    history=resnet_model.fit(train_x,train_y,30)
```

```
82/82 [==============================] - 31s 217ms/step - loss: 1.0659 - accuracy: 0.4557
```

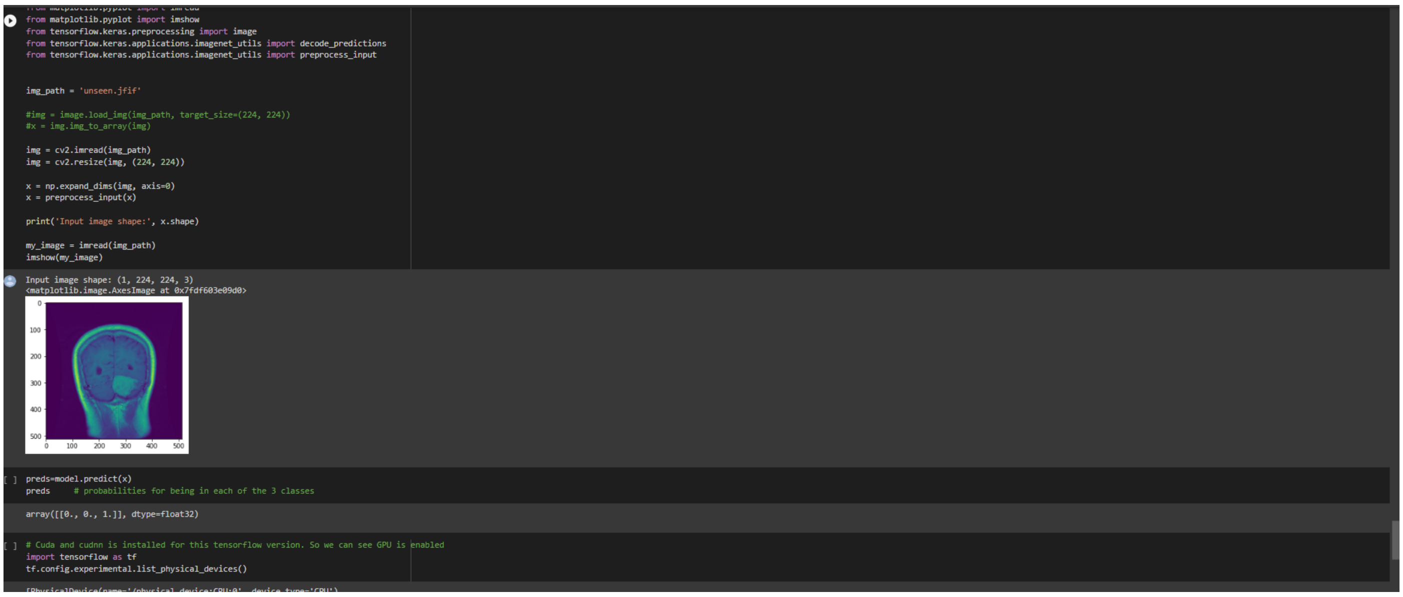
Implementing Efficientnet B0

```
[ ] from tensorflow.keras import layers
    from tensorflow.keras.applications import EfficientNetB0

    NUM_CLASSES = 3
    IMG_SIZE = 224
    size = (IMG_SIZE, IMG_SIZE)

    inputs = layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
    # Using model without transfer learning
    outputs = EfficientNetB0(include_top=True, weights=None, classes=NUM_CLASSES)(inputs)
```

B0 model starts



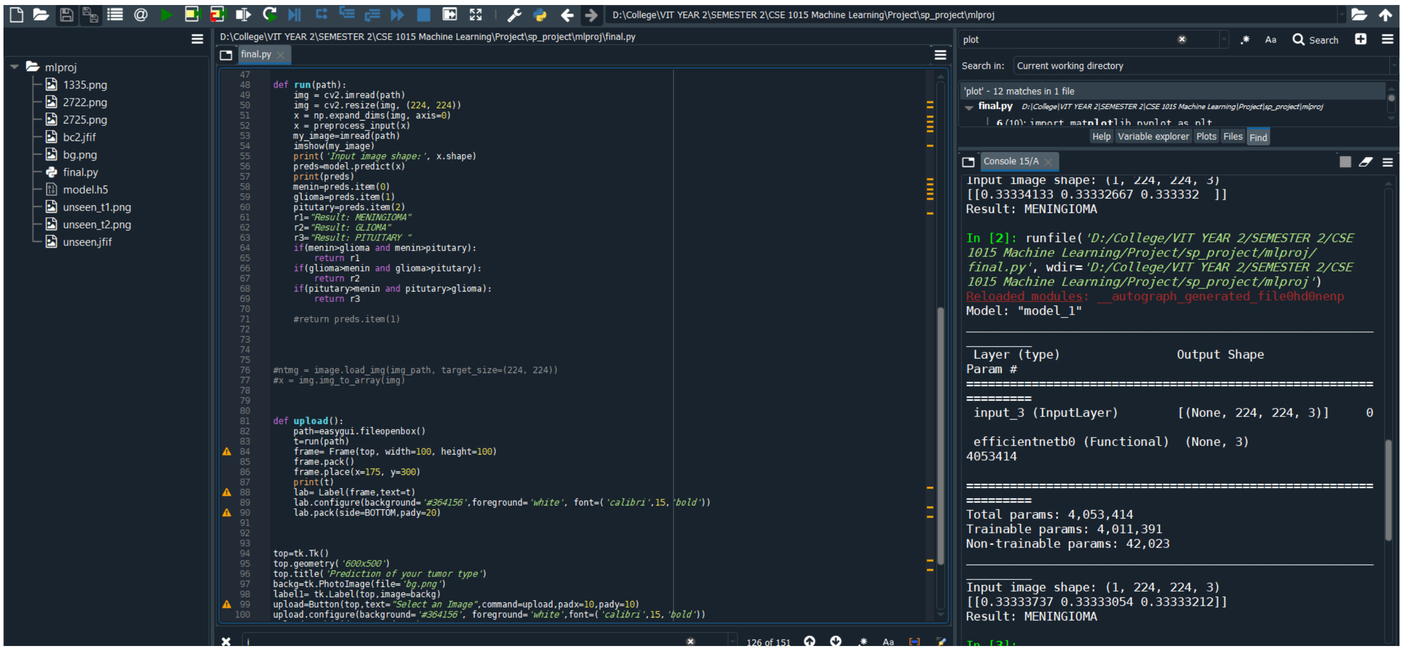
```
[ ] model = tf.keras.Model(inputs, outputs)
    model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"] )
    model.summary()

    Model: "model"
    _________________________________________________________________
     Layer (type)                Output Shape              Param #
    =================================================================
     input_1 (InputLayer)        [(None, 224, 224, 3)]     0

     efficientnetb0 (Functional)  (None, 3)                4053414

    =================================================================
    Total params: 4,053,414
    Trainable params: 4,011,391
    Non-trainable params: 42,023
    _________________________________________________________________

    hist = model.fit(train_x, train_y, epochs=30, verbose=2)

    Epoch 2/30
    77/77 - 43s - loss: 1.0880 - accuracy: 0.7201 - 43s/epoch - 552ms/step
    Epoch 3/30
    77/77 - 43s - loss: 0.8116 - accuracy: 0.7479 - 43s/epoch - 552ms/step
    Epoch 4/30
    77/77 - 43s - loss: 0.7284 - accuracy: 0.7548 - 43s/epoch - 552ms/step
    Epoch 5/30
    77/77 - 42s - loss: 0.6858 - accuracy: 0.7715 - 42s/epoch - 551ms/step
    Epoch 6/30
    77/77 - 42s - loss: 0.6238 - accuracy: 0.7809 - 42s/epoch - 550ms/step
    Epoch 7/30
    77/77 - 42s - loss: 0.5229 - accuracy: 0.7984 - 42s/epoch - 549ms/step
    Epoch 8/30
    77/77 - 43s - loss: 0.3631 - accuracy: 0.8519 - 43s/epoch - 553ms/step
    Epoch 9/30
    77/77 - 42s - loss: 0.6968 - accuracy: 0.7544 - 42s/epoch - 551ms/step
    Epoch 10/30
    77/77 - 42s - loss: 0.5660 - accuracy: 0.7760 - 42s/epoch - 551ms/step
    Epoch 11/30
    77/77 - 42s - loss: 0.3707 - accuracy: 0.8490 - 42s/epoch - 551ms/step
    Epoch 12/30
    77/77 - 43s - loss: 0.2480 - accuracy: 0.8980 - 43s/epoch - 553ms/step
```



```
from matplotlib.pyplot import imread
from matplotlib.pyplot import imshow
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.imagenet_utils import decode_predictions
from tensorflow.keras.applications.imagenet_utils import preprocess_input


img_path = 'unseen.jfif'

#img = image.load_img(img_path, target_size=(224, 224))
#x = img.img_to_array(img)

img = cv2.imread(img_path)
img = cv2.resize(img, (224, 224))

x = np.expand_dims(img, axis=0)
x = preprocess_input(x)

print('Input image shape:', x.shape)

my_image = imread(img_path)
imshow(my_image)

Input image shape: (1, 224, 224, 3)
<matplotlib.image.AxesImage at 0x7fdf603e09d0>
```

```
[ ] preds=model.predict(x)
    preds     # probabilities for being in each of the 3 classes

    array([[0., 0., 1.]], dtype=float32)

[ ] # Cuda and cudnn is installed for this tensorflow version. So we can see GPU is enabled
    import tensorflow as tf
    tf.config.experimental.list_physical_devices()

    [PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU')]
```

GUI IMPLEMENTATION:



D:\College\VIT YEAR 2\SEMESTER 2\CSE 1015 Machine Learning\Project\sp_project\mlproj

```
mlproj
    1335.png
    2722.png
    2725.png
    bc2.jfif
    bg.png
    final.py
    model.h5
    unseen_t1.png
    unseen_t2.png
    unseen.jfif
```

D:\College\VIT YEAR 2\SEMESTER 2\CSE 1015 Machine Learning\Project\sp_project\mlproj\final.py

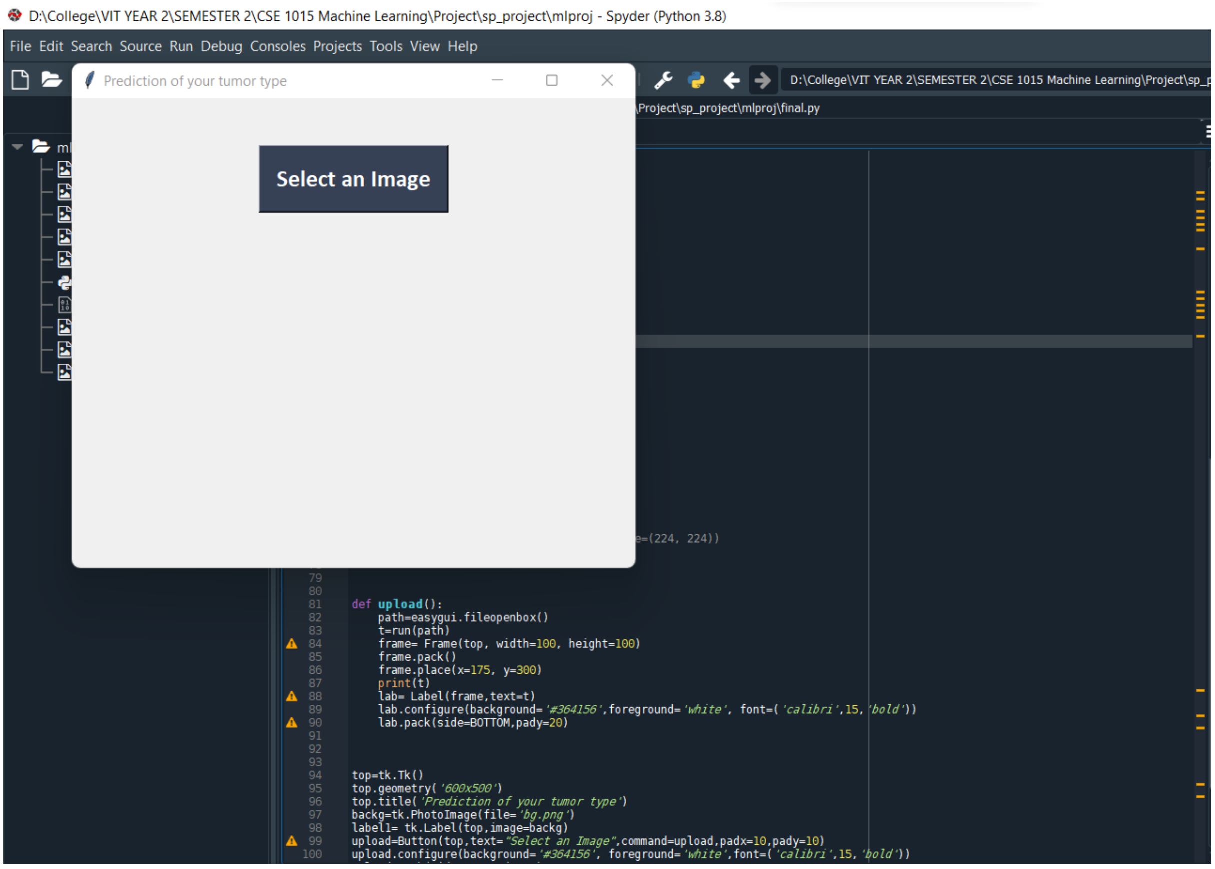
final.py

```python

def run(path):
    img = cv2.imread(path)
    img = cv2.resize(img, (224, 224))
    x = np.expand_dims(img, axis=0)
    x = preprocess_input(x)
    my_image=imread(path)
    imshow(my_image)
    print('Input image shape:', x.shape)
    preds=model.predict(x)
    print(preds)
    menin=preds.item(0)
    glioma=preds.item(1)
    pitutary=preds.item(2)
    r1="Result: MENINGIOMA"
    r2="Result: GLIOMA"
    r3="Result: PITUITARY "
    if(menin>glioma and menin>pitutary):
        return r1
    if(glioma>menin and glioma>pitutary):
        return r2
    if(pitutary>menin and pitutary>glioma):
        return r3

    #return preds.item(1)




#ntmg = image.load_img(img_path, target_size=(224, 224))
#x = img.img_to_array(img)



def upload():
    path=easygui.fileopenbox()
    t=run(path)
    frame= Frame(top, width=100, height=100)
    frame.pack()
    frame.place(x=175, y=300)
    print(t)
    lab= Label(frame,text=t)
    lab.configure(background='#364156',foreground='white', font=('calibri',15,'bold'))
    lab.pack(side=BOTTOM,pady=20)



top=tk.Tk()
top.geometry('600x500')
top.title('Prediction of your tumor type')
backg=tk.PhotoImage(file='bg.png')
label1= tk.Label(top,image=backg)
upload=Button(top,text="Select an Image",command=upload,padx=10,pady=10)
upload.configure(background='#364156', foreground='white',font=('calibri',15,'bold'))
```

```
'plot' - 12 matches in 1 file
final.py   D:|College|VIT YEAR 2|SEMESTER 2|CSE 1015 Machine Learning|Project|sp_project|mlproj
  6 (10): import matplotlib.pyplot as plt
```

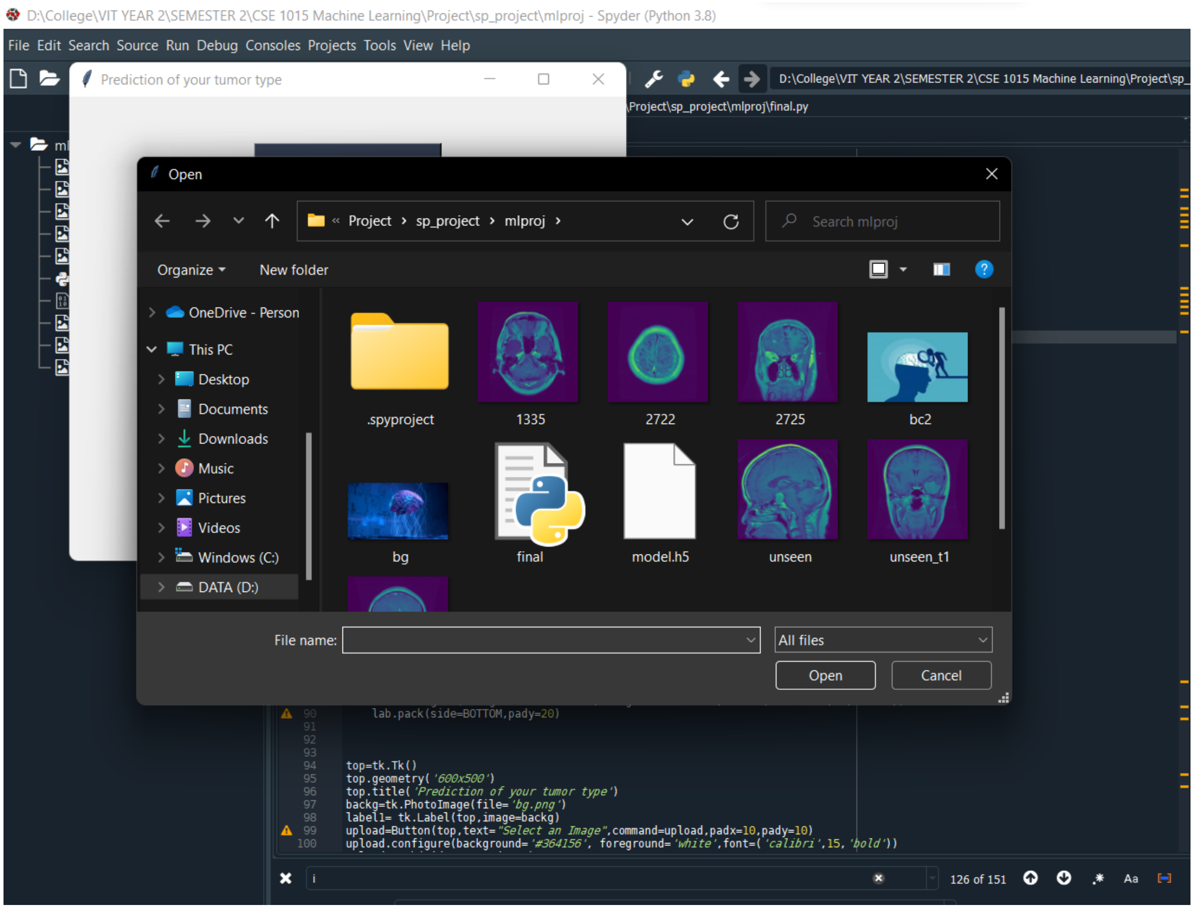
```
Input image shape: (1, 224, 224, 3)
[[0.33334133 0.33332667 0.333332  ]]
Result: MENINGIOMA

In [2]: runfile('D:/College/VIT YEAR 2/SEMESTER 2/CSE
1015 Machine Learning/Project/sp_project/mlproj/
final.py', wdir='D:/College/VIT YEAR 2/SEMESTER 2/CSE
1015 Machine Learning/Project/sp_project/mlproj')
Reloaded modules: __autograph_generated_file0hd0nenp
Model: "model_1"
________________________________________________________________________
 Layer (type)                Output Shape
Param #
========================================================================
=========
 input_3 (InputLayer)        [(None, 224, 224, 3)]     0

 efficientnetb0 (Functional)  (None, 3)
4053414

========================================================================
=========
Total params: 4,053,414
Trainable params: 4,011,391
Non-trainable params: 42,023
________________________________________________________________________
_________
Input image shape: (1, 224, 224, 3)
[[0.33333737 0.33333054 0.33333212]]
Result: MENINGIOMA

In [3]:
```

CHAPTER 4: RESULTS DISCUSSION , CONCLUSION AND FUTURE WORKS


D:\College\VIT YEAR 2\SEMESTER 2\CSE 1015 Machine Learning\Project\sp_project\mlproj - Spyder (Python 3.8)

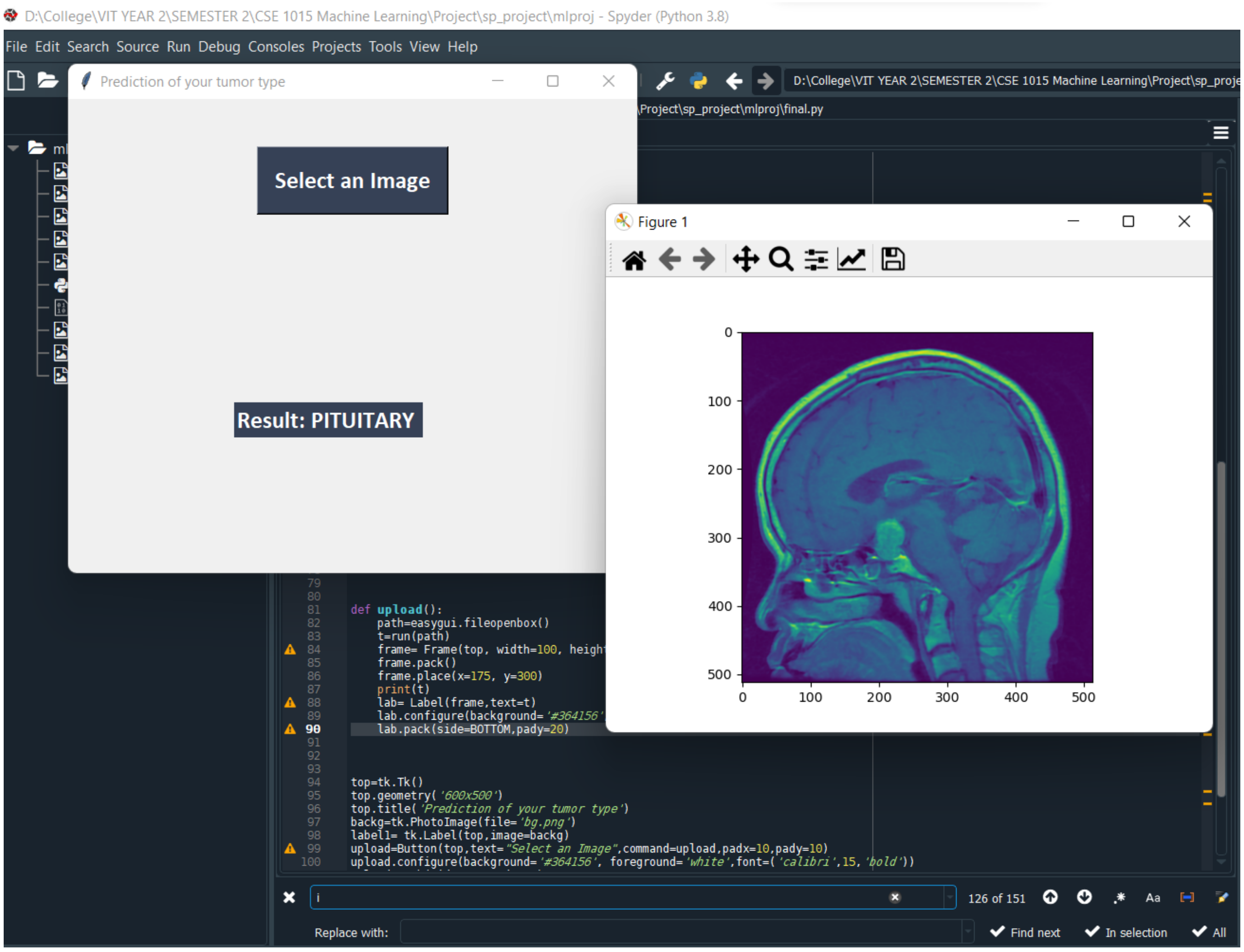
File Edit Search Source Run Debug Consoles Projects Tools View Help
Prediction of your tumor type
Select an Image
D:\College\VIT YEAR 2\SEMESTER 2\CSE 1015 Machine Learning\Project\sp_
\Project\sp_project\mlproj\final.py

```
e=(224, 224))


def upload():
    path=easygui.fileopenbox()
    t=run(path)
    frame= Frame(top, width=100, height=100)
    frame.pack()
    frame.place(x=175, y=300)
    print(t)
    lab= Label(frame,text=t)
    lab.configure(background='#364156',foreground='white', font=('calibri',15,'bold'))
    lab.pack(side=BOTTOM,pady=20)



top=tk.Tk()
top.geometry('600x500')
top.title('Prediction of your tumor type')
backg=tk.PhotoImage(file='bg.png')
labell= tk.Label(top,image=backg)
upload=Button(top,text="Select an Image",command=upload,padx=10,pady=10)
upload.configure(background='#364156', foreground='white',font=('calibri',15,'bold'))
```


D:\College\VIT YEAR 2\SEMESTER 2\CSE 1015 Machine Learning\Project\sp_project\mlproj - Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help
Prediction of your tumor type
D:\College\VIT YEAR 2\SEMESTER 2\CSE 1015 Machine Learning\Project\sp_
\Project\sp_project\mlproj\final.py
Open
« Project › sp_project › mlproj ›
Search mlproj
Organize
New folder
OneDrive - Person
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
DATA (D:)
.spyproject
1335
2722
2725
bc2
bg
final
model.h5
unseen
unseen_t1
File name:
All files
Open
Cancel
126 of 151
Aa

```
    lab.pack(side=BOTTOM,pady=20)



top=tk.Tk()
top.geometry('600x500')
top.title('Prediction of your tumor type')
backg=tk.PhotoImage(file='bg.png')
labell= tk.Label(top,image=backg)
upload=Button(top,text="Select an Image",command=upload,padx=10,pady=10)
upload.configure(background='#364156', foreground='white',font=('calibri',15,'bold'))
```


D:\College\VIT YEAR 2\SEMESTER 2\CSE 1015 Machine Learning\Project\sp_project\mlproj - Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Prediction of your tumor type
Select an Image
Result: PITUITARY
Figure 1
D:\College\VIT YEAR 2\SEMESTER 2\CSE 1015 Machine Learning\Project\sp_proje
\Project\sp_project\mlproj\final.py
126 of 151
Replace with:
Find next
In selection
All

```
def upload():
    path=easygui.fileopenbox()
    t=run(path)
    frame= Frame(top, width=100, heigh
    frame.pack()
    frame.place(x=175, y=300)
    print(t)
    lab= Label(frame,text=t)
    lab.configure(background='#364156'
    lab.pack(side=BOTTOM,pady=20)



top=tk.Tk()
top.geometry('600x500')
top.title('Prediction of your tumor type')
backg=tk.PhotoImage(file='bg.png')
label1= tk.Label(top,image=backg)
upload=Button(top,text="Select an Image",command=upload,padx=10,pady=10)
upload.configure(background='#364156', foreground='white',font=('calibri',15,'bold'))
```

Among the three models i.e VGG16 , Resnet50 and Efficient net B0, the model Efficient net B0 gives the highest accuracy then comes the Resnet50 model and then VGG16 model.

So we have implemented the weights which we got by implementing the Efficientnet B0 model in our GUI.

When a user inserted the MRI image of the Brain, the model predicts and generates the probability of the MRI image falling in the 3 different tumor classes, later based on the highest probability the model prints the type of tumor the MRI image belongs to.

Future works: For getting even more better accuracy we can implement the Efficientnet B7 Algorithm by scaling the image size and we can update the weights and further modify the GUI of the model by giving some more additional features. We can also work on a different dataset and increase the types of Brain Tumors.

REFERENCES:

Link of the Notebook:

https://colab.research.google.com/drive/1VvcXMfhhtDgO9SF8T327h8dZzfkIs6w8?usp=sharing

Link of GUI:

https://drive.google.com/drive/folders/16p3_rIEsVCTfmhA0VC6UhpjWtC56XZTB?usp=sharing

Reference Links:

https://www.topbots.com/important-cnn-architectures/

https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

https://www.researchgate.net/figure/Illustrations-of-three-typical-brain-tumors-a-meningioma-b-glioma-and-c_fig1_307649824

https://dl.acm.org/doi/abs/10.1145/2222444.2222467

https://ieeexplore.ieee.org/abstract/document/8964846

https://ieeexplore.ieee.org/abstract/document/6020885