สมาชิก

กิตติธัช ฉายาหทัย 6410210030

สันเพ็ชร แซ่ฟัง 6410210319

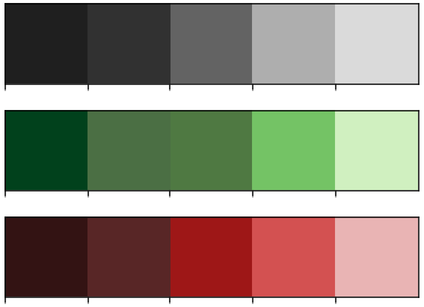
```
import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'brain-tumor-classification-mri:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F672377%2F1183165%2Fbundle%2Farchive.zip%3FX-Goog-Algorithm%3DGOOG4-RSA-SHA256%

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')
```

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import cv2
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tqdm import tqdm
import os
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, TensorBoard, ModelCheckpoint
from sklearn.metrics import classification_report,confusion_matrix
import ipywidgets as widgets
import io
from PIL import Image
from IPython.display import display,clear_output
from warnings import filterwarnings
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(84).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(44).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(245).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/6.jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(238).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(196).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(108).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(310).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image (5).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(186).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(29).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(140).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(224).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image (61).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(173).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(52).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image (52).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(174).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(203).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(33).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(283).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(291).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(243).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(284).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(106).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/5.jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(69).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(302).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/8.jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(21).jpg
```

```
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(258).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(103).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(265).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(289).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(278).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(215).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(155).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(184).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image (40).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image (26).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(193).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(74).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(35).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(6).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(15).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(132).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(135).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image (23).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(168).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(81).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(204).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(207).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(13).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(293).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(226).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(123).jpg
/kaggle/input/brain-tumor-classification-mri/Training/no_tumor/image(237).jpg
```

```
colors_dark = ["#1F1F1F", "#313131", '#636363', '#AEAEAE', '#DADADA']
colors_red = ["#331313", "#582626", '#9E1717', '#D35151', '#E9B4B4']
colors_green = ['#01411C','#4B6F44','#4F7942','#74C365','#D0F0C0']

sns.palplot(colors_dark)
sns.palplot(colors_green)
sns.palplot(colors_red)
```

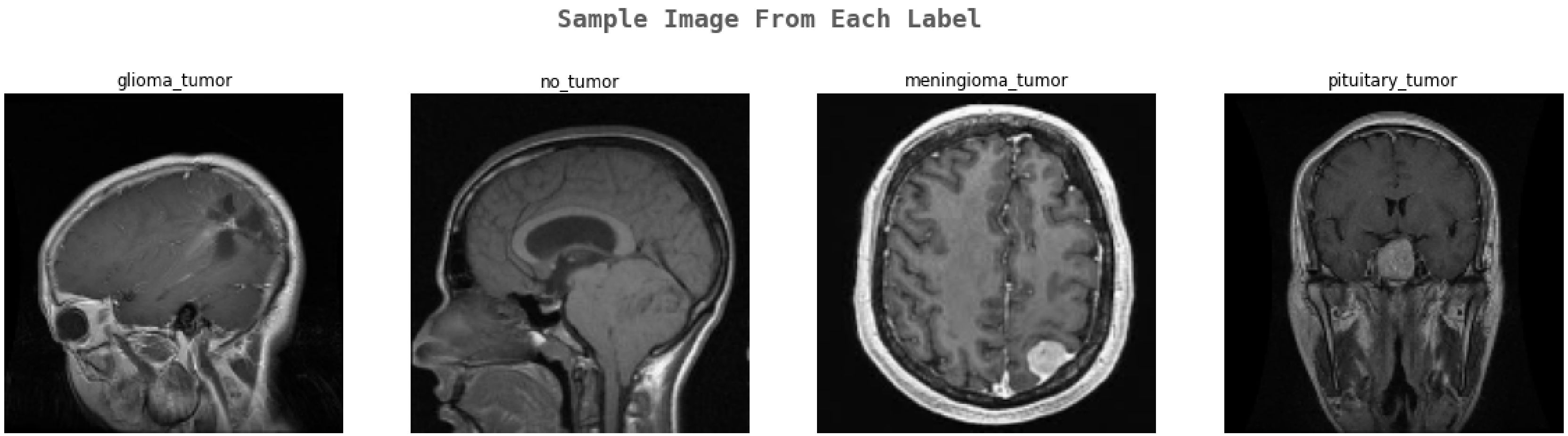


```
labels = ['glioma_tumor','no_tumor','meningioma_tumor','pituitary_tumor']
```

```
X_train = []
y_train = []
image_size = 150
for i in labels:
    folderPath = os.path.join('../input/brain-tumor-classification-mri','Training',i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath,j))
        img = cv2.resize(img,(image_size, image_size))
        X_train.append(img)
        y_train.append(i)

for i in labels:
    folderPath = os.path.join('../input/brain-tumor-classification-mri','Testing',i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath,j))
        img = cv2.resize(img,(image_size,image_size))
        X_train.append(img)
        y_train.append(i)

X_train = np.array(X_train)
y_train = np.array(y_train)
```

```
100%|██████████| 826/826 [00:08<00:00, 99.70it/s]
100%|██████████| 395/395 [00:03<00:00, 105.83it/s]
100%|██████████| 822/822 [00:08<00:00, 99.88it/s]
100%|██████████| 827/827 [00:08<00:00, 95.88it/s]
100%|██████████| 100/100 [00:00<00:00, 100.78it/s]
100%|██████████| 105/105 [00:00<00:00, 161.43it/s]
100%|██████████| 115/115 [00:00<00:00, 126.92it/s]
100%|██████████| 74/74 [00:00<00:00, 75.54it/s]
```

```
k=0
fig, ax = plt.subplots(1,4,figsize=(20,20))
fig.text(s='Sample Image From Each Label',size=18,fontweight='bold',
             fontname='monospace',color=colors_dark[1],y=0.62,x=0.4,alpha=0.8)
for i in labels:
    j=0
    while True :
        if y_train[j]==i:
            ax[k].imshow(X_train[j])
            ax[k].set_title(y_train[j])
            ax[k].axis('off')
            k+=1
            break
        j+=1
```



```
X_train, y_train = shuffle(X_train,y_train, random_state=101)
```

```
X_train.shape
```

```
    (3264, 150, 150, 3)
```

```
X_train,X_test,y_train,y_test = train_test_split(X_train,y_train, test_size=0.1,random_state=101)
```

```
y_train_new = []
for i in y_train:
    y_train_new.append(labels.index(i))
y_train = y_train_new
y_train = tf.keras.utils.to_categorical(y_train)
```

```
y_test_new = []
for i in y_test:
    y_test_new.append(labels.index(i))
y_test = y_test_new
y_test = tf.keras.utils.to_categorical(y_test)
```

```
effnet = EfficientNetB0(weights='imagenet',include_top=False,input_shape=(image_size,image_size,3))
```

```
    Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb0_notop.h5
    16711680/16705208 [==============================] - 0s 0us/step
```

```
model = effnet.output
model = tf.keras.layers.GlobalAveragePooling2D()(model)
model = tf.keras.layers.Dropout(rate=0.5)(model)
model = tf.keras.layers.Dense(4,activation='softmax')(model)
model = tf.keras.models.Model(inputs=effnet.input, outputs = model)
```

```
model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            [(None, 150, 150, 3) 0
__________________________________________________________________________________________________
rescaling_1 (Rescaling)         (None, 150, 150, 3)  0           input_2[0][0]
__________________________________________________________________________________________________
normalization_1 (Normalization) (None, 150, 150, 3)  7           rescaling_1[0][0]
__________________________________________________________________________________________________
stem_conv_pad (ZeroPadding2D)   (None, 151, 151, 3)  0           normalization_1[0][0]
__________________________________________________________________________________________________
stem_conv (Conv2D)              (None, 75, 75, 32)   864         stem_conv_pad[0][0]
__________________________________________________________________________________________________
stem_bn (BatchNormalization)    (None, 75, 75, 32)   128         stem_conv[0][0]
__________________________________________________________________________________________________
stem_activation (Activation)    (None, 75, 75, 32)   0           stem_bn[0][0]
__________________________________________________________________________________________________
block1a_dwconv (DepthwiseConv2D (None, 75, 75, 32)   288         stem_activation[0][0]
__________________________________________________________________________________________________
block1a_bn (BatchNormalization) (None, 75, 75, 32)   128         block1a_dwconv[0][0]
__________________________________________________________________________________________________
block1a_activation (Activation) (None, 75, 75, 32)   0           block1a_bn[0][0]
__________________________________________________________________________________________________
block1a_se_squeeze (GlobalAvera (None, 32)           0           block1a_activation[0][0]
__________________________________________________________________________________________________
block1a_se_reshape (Reshape)    (None, 1, 1, 32)     0           block1a_se_squeeze[0][0]
__________________________________________________________________________________________________
block1a_se_reduce (Conv2D)      (None, 1, 1, 8)      264         block1a_se_reshape[0][0]
__________________________________________________________________________________________________
block1a_se_expand (Conv2D)      (None, 1, 1, 32)     288         block1a_se_reduce[0][0]
__________________________________________________________________________________________________
block1a_se_excite (Multiply)    (None, 75, 75, 32)   0           block1a_activation[0][0]
                                                                 block1a_se_expand[0][0]
__________________________________________________________________________________________________
block1a_project_conv (Conv2D)   (None, 75, 75, 16)   512         block1a_se_excite[0][0]
__________________________________________________________________________________________________
block1a_project_bn (BatchNormal (None, 75, 75, 16)   64          block1a_project_conv[0][0]
__________________________________________________________________________________________________
block2a_expand_conv (Conv2D)    (None, 75, 75, 96)   1536        block1a_project_bn[0][0]
__________________________________________________________________________________________________
block2a_expand_bn (BatchNormali (None, 75, 75, 96)   384         block2a_expand_conv[0][0]
__________________________________________________________________________________________________
block2a_expand_activation (Acti (None, 75, 75, 96)   0           block2a_expand_bn[0][0]
__________________________________________________________________________________________________
block2a_dwconv_pad (ZeroPadding (None, 77, 77, 96)   0           block2a_expand_activation[0][0]
block2a_dwconv (DepthwiseConv2D (None, 38, 38, 96)   864         block2a_dwconv_pad[0][0]

block2a_bn (BatchNormalization) (None, 38, 38, 96)   384         block2a_dwconv[0][0]

block2a_activation (Activation) (None, 38, 38, 96)   0           block2a_bn[0][0]

block2a_se_squeeze (GlobalAvera (None, 96)           0           block2a_activation[0][0]

block2a_se_reshape (Reshape)    (None, 1, 1, 96)     0           block2a_se_squeeze[0][0]

block2a_se_reduce (Conv2D)      (None, 1, 1, 4)      388         block2a_se_reshape[0][0]
```

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

```
tensorboard = TensorBoard(log_dir = 'logs')
checkpoint = ModelCheckpoint("effnet.h5",monitor="val_accuracy",save_best_only=True,mode="auto",verbose=1)
reduce_lr = ReduceLROnPlateau(monitor = 'val_accuracy', factor = 0.3, patience = 2, min_delta = 0.001,
                              mode='auto',verbose=1)
```

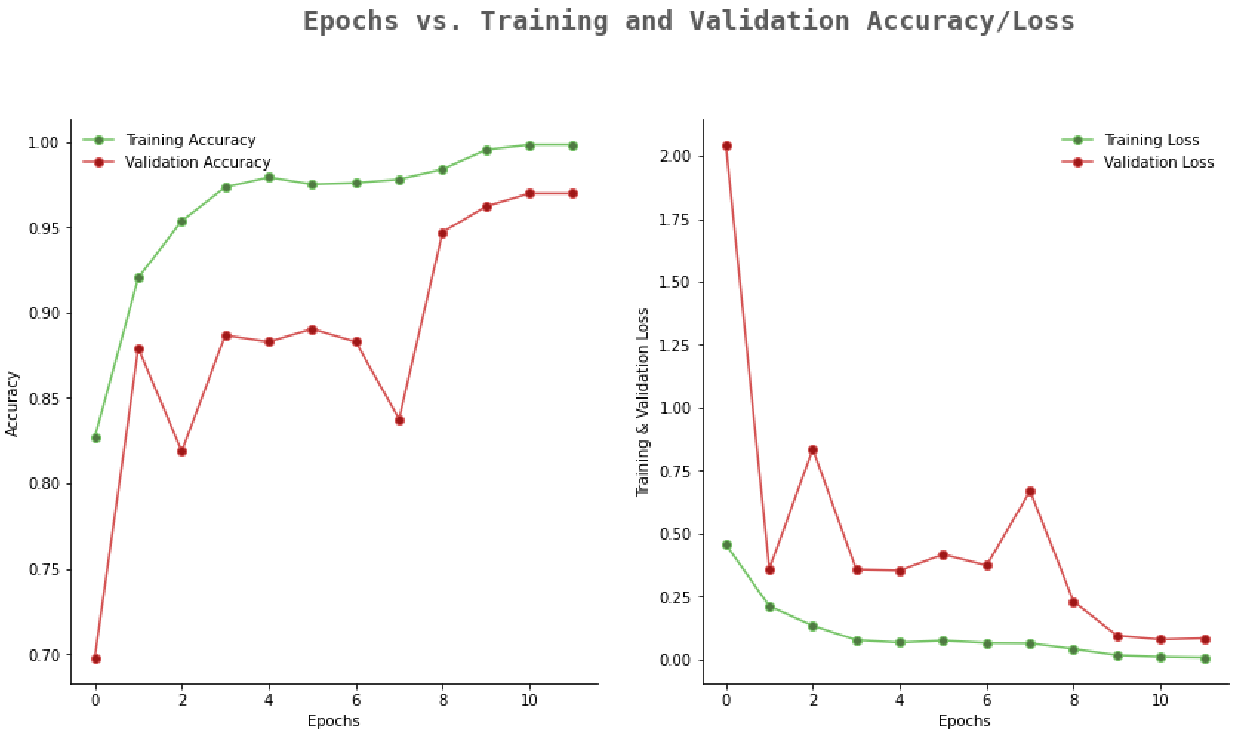
```
history = model.fit(X_train,y_train,validation_split=0.1, epochs =12, verbose=1, batch_size=32,
                    callbacks=[tensorboard,checkpoint,reduce_lr])
```

```
Epoch 1/12
75/75 [==============================] - 24s 152ms/step - loss: 0.6863 - accuracy: 0.7187 - val_loss: 2.0442 - val_accuracy: 0.6981

Epoch 00001: val_accuracy improved from -inf to 0.69811, saving model to effnet.h5
Epoch 2/12
75/75 [==============================] - 8s 112ms/step - loss: 0.2298 - accuracy: 0.9133 - val_loss: 0.3594 - val_accuracy: 0.8792

Epoch 00002: val_accuracy improved from 0.69811 to 0.87925, saving model to effnet.h5
Epoch 3/12
75/75 [==============================] - 8s 111ms/step - loss: 0.1471 - accuracy: 0.9494 - val_loss: 0.8350 - val_accuracy: 0.8189

Epoch 00003: val_accuracy did not improve from 0.87925
Epoch 4/12
75/75 [==============================] - 8s 112ms/step - loss: 0.0903 - accuracy: 0.9675 - val_loss: 0.3573 - val_accuracy: 0.8868

Epoch 00004: val_accuracy improved from 0.87925 to 0.88679, saving model to effnet.h5
Epoch 5/12
75/75 [==============================] - 8s 112ms/step - loss: 0.0651 - accuracy: 0.9749 - val_loss: 0.3526 - val_accuracy: 0.8830

Epoch 00005: val_accuracy did not improve from 0.88679
Epoch 6/12
75/75 [==============================] - 8s 112ms/step - loss: 0.0820 - accuracy: 0.9717 - val_loss: 0.4170 - val_accuracy: 0.8906

Epoch 00006: val_accuracy improved from 0.88679 to 0.89057, saving model to effnet.h5
Epoch 7/12
75/75 [==============================] - 8s 111ms/step - loss: 0.0613 - accuracy: 0.9761 - val_loss: 0.3729 - val_accuracy: 0.8830
```

```
Epoch 00007: val_accuracy did not improve from 0.89057
Epoch 8/12
75/75 [==============================] - 8s 112ms/step - loss: 0.0537 - accuracy: 0.9806 - val_loss: 0.6694 - val_accuracy: 0.8377

Epoch 00008: val_accuracy did not improve from 0.89057

Epoch 00008: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
Epoch 9/12
75/75 [==============================] - 8s 112ms/step - loss: 0.0458 - accuracy: 0.9806 - val_loss: 0.2302 - val_accuracy: 0.9472

Epoch 00009: val_accuracy improved from 0.89057 to 0.94717, saving model to effnet.h5
Epoch 10/12
75/75 [==============================] - 8s 112ms/step - loss: 0.0148 - accuracy: 0.9966 - val_loss: 0.0945 - val_accuracy: 0.9623

Epoch 00010: val_accuracy improved from 0.94717 to 0.96226, saving model to effnet.h5
Epoch 11/12
75/75 [==============================] - 8s 112ms/step - loss: 0.0103 - accuracy: 0.9979 - val_loss: 0.0788 - val_accuracy: 0.9698

Epoch 00011: val_accuracy improved from 0.96226 to 0.96981, saving model to effnet.h5
Epoch 12/12
75/75 [==============================] - 8s 112ms/step - loss: 0.0098 - accuracy: 0.9983 - val_loss: 0.0838 - val_accuracy: 0.9698

Epoch 00012: val_accuracy did not improve from 0.96981
```

```
filterwarnings('ignore')

epochs = [i for i in range(12)]
fig, ax = plt.subplots(1,2,figsize=(14,7))
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']

fig.text(s='Epochs vs. Training and Validation Accuracy/Loss',size=18,fontweight='bold',
            fontname='monospace',color=colors_dark[1],y=1,x=0.28,alpha=0.8)

sns.despine()
ax[0].plot(epochs, train_acc, marker='o',markerfacecolor=colors_green[2],color=colors_green[3],
           label = 'Training Accuracy')
ax[0].plot(epochs, val_acc, marker='o',markerfacecolor=colors_red[2],color=colors_red[3],
           label = 'Validation Accuracy')
ax[0].legend(frameon=False)
ax[0].set_xlabel('Epochs')
ax[0].set_ylabel('Accuracy')

sns.despine()
ax[1].plot(epochs, train_loss, marker='o',markerfacecolor=colors_green[2],color=colors_green[3],
           label ='Training Loss')
ax[1].plot(epochs, val_loss, marker='o',markerfacecolor=colors_red[2],color=colors_red[3],
           label = 'Validation Loss')
ax[1].legend(frameon=False)
ax[1].set_xlabel('Epochs')
ax[1].set_ylabel('Training & Validation Loss')
```

```
ax[1].set_ylabel('Training & Validation Loss')

fig.show()
```

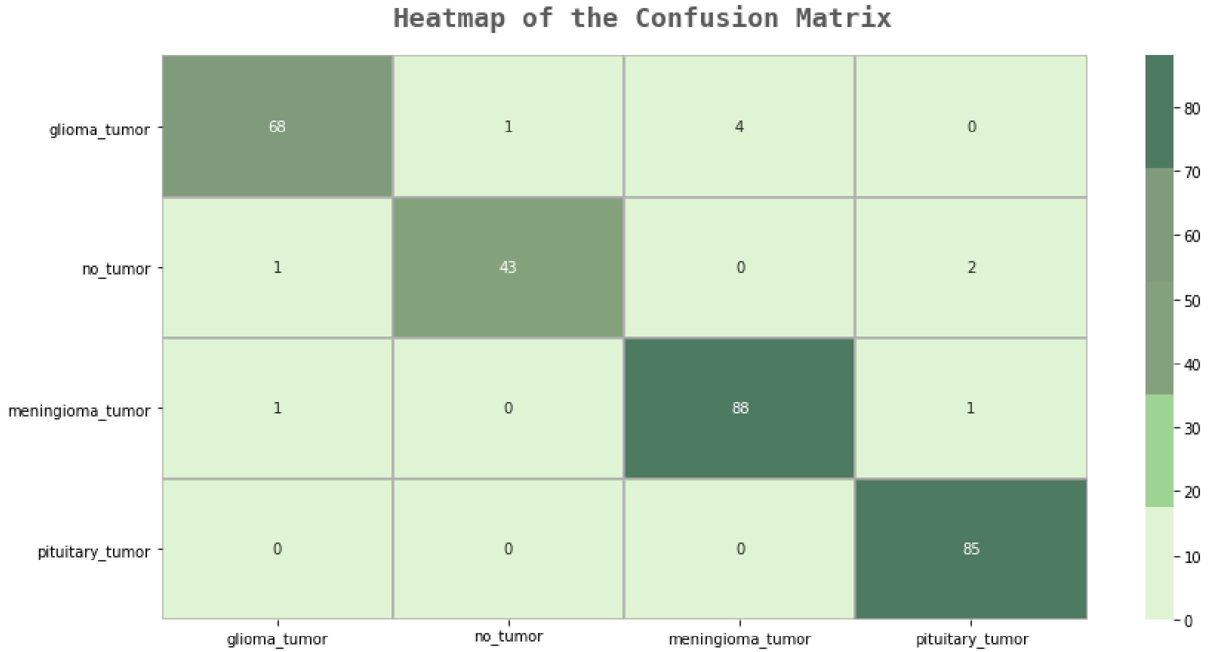


```
pred = model.predict(X_test)
pred = np.argmax(pred,axis=1)
y_test_new = np.argmax(y_test,axis=1)
```

```
print(classification_report(y_test_new,pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.93      0.95        73
           1       0.98      0.93      0.96        46
           2       0.96      0.98      0.97        90
           3       0.97      1.00      0.98        85

    accuracy                           0.97       294
   macro avg       0.97      0.96      0.96       294
weighted avg       0.97      0.97      0.97       294
```

```
fig,ax=plt.subplots(1,1,figsize=(14,7))
sns.heatmap(confusion_matrix(y_test_new,pred),ax=ax,xticklabels=labels,yticklabels=labels,annot=True,
           cmap=colors_green[::-1],alpha=0.7,linewidths=2,linecolor=colors_dark[3])
fig.text(s='Heatmap of the Confusion Matrix',size=18,fontweight='bold',
             fontname='monospace',color=colors_dark[1],y=0.92,x=0.28,alpha=0.8)

plt.show()
```



```
def img_pred(upload):
    for name, file_info in uploader.value.items():
        img = Image.open(io.BytesIO(file_info['content']))

uploader = widgets.FileUpload()
display(uploader)
```

```
FileUpload(value={}, description='Upload')
```

```
button = widgets.Button(description='Predict')
out = widgets.Output()
def on_button_clicked(_):
    with out:
        clear_output()
        try:
            img_pred(uploader)

        except:
            print('No Image Uploaded/Invalid Image File')
button.on_click(on_button_clicked)
widgets.VBox([button,out])
```

```
VBox(children=(Button(description='Predict', style=ButtonStyle()), Output()))
```

ดับเบิลคลิก (หรือกด Enter) เพื่อแก้ไข

ดับเบิลคลิก (หรือกด Enter) เพื่อแก้ไข

ดับเบิลคลิก (หรือกด Enter) เพื่อแก้ไข

ดับเบิลคลิก (หรือกด Enter) เพื่อแก้ไข