**MRI Brain Tumor Classification:**

**Source Code:**

**By:**

**Raghav Jain – 219303166**

**Sivam Pratik – 219303066**

import numpy as np

import pandas as pd

import os

import cv2

import matplotlib.pyplot as plt

import seaborn as sns

import tensorflow as tf

import keras

from tqdm import tqdm

from keras.callbacks import EarlyStopping,ModelCheckpoint

from sklearn.metrics import confusion\_matrix , accuracy\_score

from sklearn.metrics import classification\_report

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

import glob

import pandas as pan

import matplotlib.pyplot as plotter

import warnings

warnings.filterwarnings('ignore')

image\_data='/content/drive/MyDrive/brain\_mri\_scan\_images'

pd.DataFrame(os.listdir(image\_data),columns=['Files\_Name'])

files = [i for i in glob.glob(image\_data + "//\*//\*")]

np.random.shuffle(files)

labels = [os.path.dirname(i).split("/")[-1] for i in files]

data = zip(files, labels)

dataframe = pan.DataFrame(data, columns = ["Image", "Label"])

dataframe

sns.countplot(x = dataframe["Label"])

plotter.xticks(rotation = 50);

train\_data\_dir =image\_data

batch\_size = 32

target\_size = (224,224)

validation\_split = 0.2

train= tf.keras.preprocessing.image\_dataset\_from\_directory(

    train\_data\_dir,

    validation\_split=validation\_split,

    subset="training",

    seed=100,

    image\_size=target\_size,

    batch\_size=batch\_size,

)

validation= tf.keras.preprocessing.image\_dataset\_from\_directory(

    train\_data\_dir,

    validation\_split=validation\_split,

    subset="validation",

    seed=200,

    image\_size=target\_size,

    batch\_size=batch\_size,

)

class\_names = train.class\_names

class\_names

plt.figure(figsize=(15, 20))

for images, labels in train.take(1):

    for i in range(8):

        ax = plt.subplot(8, 4, i + 1)

        plt.imshow(images[i].numpy().astype("uint8"))

        plt.title(class\_names[labels[i]])

        plt.axis("off")

base\_model = tf.keras.applications.EfficientNetV2B0(input\_shape=(224,224,3),include\_top=False,weights='imagenet')

base\_model.trainable = False

keras\_model=keras.models.Sequential()

keras\_model.add(base\_model)

keras\_model.add(keras.layers.Flatten())

keras\_model.add(keras.layers.Dropout(0.5))

keras\_model.add(keras.layers.Dense(10,activation=tf.nn.softmax))

keras\_model.summary()

tf.keras.utils.plot\_model(keras\_model, to\_file='model.png', show\_shapes=True, show\_layer\_names=True,show\_dtype=True,dpi=80)

checkpoint =ModelCheckpoint("my\_keras\_model.h5", save\_best\_only=True)

early\_stopping =EarlyStopping(patience=5, restore\_best\_weights=True)

keras\_model.compile(optimizer ='Adam',loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

hist=keras\_model.fit\_generator(train,epochs=20,validation\_data=validation,callbacks=[checkpoint,early\_stopping])

score, acc = keras\_model.evaluate(validation)

print('Test Loss =', score)

print('Test Accuracy =', acc)

hist\_=pd.DataFrame(hist.history)

hist\_

plt.figure(figsize=(15,5))

plt.subplot(1,2,1)

plt.plot(hist\_['loss'],label='Train\_Loss')

plt.plot(hist\_['val\_loss'],label='Validation\_Loss')

plt.title('Train\_Loss & Validation\_Loss',fontsize=20)

plt.legend()

plt.subplot(1,2,2)

plt.plot(hist\_['accuracy'],label='Train\_Accuracy')

plt.plot(hist\_['val\_accuracy'],label='Validation\_Accuracy')

plt.title('Train\_Accuracy & Validation\_Accuracy',fontsize=20)

plt.legend()

X\_val,y\_val,y\_pred=[],[],[]

for images, labels in validation:

    y\_val.extend(labels.numpy())

    X\_val.extend(images.numpy())

predictions=keras\_model.predict(np.array(X\_val))

for i in predictions:

    y\_pred.append(np.argmax(i))

df=pd.DataFrame()

df['Actual'],df['Prediction']=y\_val,y\_pred

df

plt.figure(figsize=(25,25))

for i in range(32):

    ax = plt.subplot(8, 4, i + 1)

    plt.imshow(X\_val[i].astype("uint8"))

    plt.title(f'{class\_names[y\_val[i]]} :: {class\_names[y\_pred[i]]}')

    plt.axis("off")

ax= plt.subplot()

CM = confusion\_matrix(y\_val,y\_pred)

sns.heatmap(CM, annot=True, fmt='g', ax=ax,cbar=False,cmap='RdBu')

ax.set\_xlabel('Predicted labels')

ax.set\_ylabel('True labels')

ax.set\_title('Confusion Matrix')

plt.show()

CM