```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.5.16)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from kaggle) (2024.2.2)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.66.2)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.0.7)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle) (6.1.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.6)
```

```
# configuring the path of Kaggle.json file
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d shreyag1103/brain-mri-scans-for-brain-tumor-classification
```

```
brain-mri-scans-for-brain-tumor-classification.zip: Skipping, found more recently modified local copy (use --force to force downloac
```

```
!ls
```

```
brain-mri-scans-for-brain-tumor-classification.zip  data  kaggle.json  sample_data
```

```
#extract the file
from zipfile import ZipFile

# dataset='bone-fracture-detection-using-xrays.zip'

dataset = '/content/brain-mri-scans-for-brain-tumor-classification.zip'

with ZipFile(dataset, 'r') as zip:
  zip.extractall()
  print('The dataset is extracted')
```

```
The dataset is extracted
```

```
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
from google.colab.patches import cv2_imshow
from PIL import Image
from sklearn.model_selection import train_test_split
```

```
tumor_glioma_files=os.listdir('/content/data/glioma')
```

```
tumor_meningioma_files=os.listdir('/content/data/meningioma')
```

```
tumor_pituitary_files=os.listdir('/content/data/pituitary')
```

```
non_tumor_files=os.listdir('/content/data/notumor')
```

```
print('Number of with glioma images:', len(tumor_glioma_files))
print('Number of with meningioma images:', len(tumor_meningioma_files))
print('Number of with pituitary images:', len(tumor_pituitary_files))
print('Number of non tumor images:', len(non_tumor_files))
```

```
Number of with glioma images: 300
Number of with meningioma images: 306
Number of with pituitary images: 300
Number of non tumor images: 405
```

Creating Labels for the two class of Images

glioma --> 1 meningioma --> 2 pituitary --> 3 non tumor --> 0

```
# create the labels

tumor_glioma_labels = [1]*300

tumor_meningioma_labels = [2]*306

tumor_pituitary_labels = [3]*300

non_tumor_labels = [0]*405



print(tumor_glioma_labels[0:5])

print(tumor_meningioma_labels[0:5])

print(tumor_pituitary_labels[0:5])

print(non_tumor_labels[0:5])
```

```
[1, 1, 1, 1, 1]
[2, 2, 2, 2, 2]
[3, 3, 3, 3, 3]
[0, 0, 0, 0, 0]
```

```
print(len(tumor_glioma_labels))

print(len(tumor_meningioma_labels))

print(len(tumor_pituitary_labels))

print(len(non_tumor_labels))
```

```
300
306
300
405
```

```
labels = tumor_glioma_labels+tumor_meningioma_labels+tumor_pituitary_labels+non_tumor_labels

print(len(labels))
print(labels[0:5])
print(labels[-5:])
```

```
1311
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
```

Image Processing

Resize the Images

Convert the images to numpy arrays

```
# convert images to numpy arrays+
data=[]

tumor_glioma_path='/content/data/glioma/'

for img_file in tumor_glioma_files:

  image = Image.open(tumor_glioma_path + img_file)
  image = image.resize((128,128))
  image = image.convert('RGB')
  image = np.array(image)
  data.append(image)

tumor_meningioma_path='/content/data/meningioma/'

for img_file in tumor_meningioma_files:

  image = Image.open(tumor_meningioma_path + img_file)
  image = image.resize((128,128))
  image = image.convert('RGB')
  image = np.array(image)
  data.append(image)

tumor_pituitary_path='/content/data/pituitary/'

for img_file in tumor_pituitary_files:

  image = Image.open(tumor_pituitary_path + img_file)
  image = image.resize((128,128))
  image = image.convert('RGB')
  image = np.array(image)
  data.append(image)

non_tumor_path='/content/data/notumor/'

for img_file in non_tumor_files:

  image = Image.open(non_tumor_path + img_file)
  image = image.resize((128,128))
  image = image.convert('RGB')
  image = np.array(image)
  data.append(image)
```

Double-click (or enter) to edit

```
type(data)
```

```
list
```

```
len(data)
```

```
1311
```

```
data[0]
```

```
ndarray (128, 128, 3) show data
```

```
type(data[0])
```

```
numpy.ndarray
```

```
data[0].shape
```

```
(128, 128, 3)
```

```
# converting image list and label list to numpy arrays

X = np.array(data)
Y = np.array(labels)
```

```
type(X)
```

```
numpy.ndarray
```

```
type(Y)
```

```
numpy.ndarray
```

```
print(X.shape)
print(Y.shape)
```

```
(1311, 128, 128, 3)
(1311,)
```

```
print(Y)
```

```
[1 1 1 ... 0 0 0]
```

Train Test Split

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

```
# scaling the data

X_train_scaled = X_train/255

X_test_scaled = X_test/255
```

Building a Convolutional Neural Networks (CNN)

```
import tensorflow as tf
from tensorflow import keras
```

```
num_of_classes = 4

model = keras.Sequential()

model.add(keras.layers.Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=(128,128,3)))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))


model.add(keras.layers.Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))

model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(128, activation='relu'))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dropout(0.5))


model.add(keras.layers.Dense(num_of_classes, activation='sigmoid'))
```

```
# compile the neural network
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['acc'])
```

```
# training the neural network
history = model.fit(X_train_scaled, Y_train, validation_split=0.1, epochs=50)
```

```
30/30 [==============================] - 23s 745ms/step - loss: 0.0174 - acc: 0.9926 - val_loss: 1.2492 - val_acc: 0.8286
Epoch 29/50
30/30 [==============================] - 24s 786ms/step - loss: 0.0200 - acc: 0.9905 - val_loss: 1.2380 - val_acc: 0.8190
Epoch 30/50
30/30 [==============================] - 24s 788ms/step - loss: 0.0335 - acc: 0.9894 - val_loss: 1.4088 - val_acc: 0.7905
Epoch 31/50
30/30 [==============================] - 25s 823ms/step - loss: 0.0331 - acc: 0.9841 - val_loss: 1.3083 - val_acc: 0.8190
Epoch 32/50
30/30 [==============================] - 24s 793ms/step - loss: 0.0224 - acc: 0.9894 - val_loss: 1.5674 - val_acc: 0.8095
Epoch 33/50
30/30 [==============================] - 22s 742ms/step - loss: 0.0290 - acc: 0.9926 - val_loss: 1.1468 - val_acc: 0.8286
Epoch 34/50
30/30 [==============================] - 23s 765ms/step - loss: 0.0177 - acc: 0.9936 - val_loss: 1.5623 - val_acc: 0.8095
Epoch 35/50
30/30 [==============================] - 24s 801ms/step - loss: 0.0348 - acc: 0.9905 - val_loss: 1.2125 - val_acc: 0.8095
Epoch 36/50
30/30 [==============================] - 24s 808ms/step - loss: 0.0127 - acc: 0.9968 - val_loss: 1.2045 - val_acc: 0.8286
Epoch 37/50
30/30 [==============================] - 24s 801ms/step - loss: 0.0118 - acc: 0.9947 - val_loss: 1.3269 - val_acc: 0.8095
Epoch 38/50
30/30 [==============================] - 24s 799ms/step - loss: 0.0125 - acc: 0.9958 - val_loss: 1.3687 - val_acc: 0.8190
Epoch 39/50
30/30 [==============================] - 24s 818ms/step - loss: 0.0177 - acc: 0.9905 - val_loss: 1.3763 - val_acc: 0.8381
Epoch 40/50
30/30 [==============================] - 23s 747ms/step - loss: 0.0091 - acc: 0.9979 - val_loss: 1.7904 - val_acc: 0.8095
Epoch 41/50
30/30 [==============================] - 23s 776ms/step - loss: 0.0236 - acc: 0.9926 - val_loss: 1.3823 - val_acc: 0.8286
Epoch 42/50
30/30 [==============================] - 23s 778ms/step - loss: 0.0351 - acc: 0.9915 - val_loss: 1.3322 - val_acc: 0.7905
Epoch 43/50
30/30 [==============================] - 24s 790ms/step - loss: 0.0346 - acc: 0.9883 - val_loss: 1.3025 - val_acc: 0.8000
Epoch 44/50
30/30 [==============================] - 23s 784ms/step - loss: 0.0171 - acc: 0.9926 - val_loss: 1.2454 - val_acc: 0.8000
Epoch 45/50
30/30 [==============================] - 23s 752ms/step - loss: 0.0363 - acc: 0.9830 - val_loss: 1.2533 - val_acc: 0.8000
Epoch 46/50
30/30 [==============================] - 24s 791ms/step - loss: 0.0154 - acc: 0.9926 - val_loss: 1.3039 - val_acc: 0.8000
Epoch 47/50
30/30 [==============================] - 24s 815ms/step - loss: 0.0219 - acc: 0.9926 - val_loss: 1.3934 - val_acc: 0.8286
Epoch 48/50
30/30 [==============================] - 25s 815ms/step - loss: 0.0141 - acc: 0.9936 - val_loss: 1.4125 - val_acc: 0.8190
Epoch 49/50
30/30 [==============================] - 24s 796ms/step - loss: 0.0164 - acc: 0.9936 - val_loss: 1.7907 - val_acc: 0.8095
Epoch 50/50
30/30 [==============================] - 23s 780ms/step - loss: 0.0310 - acc: 0.9905 - val_loss: 1.3535 - val_acc: 0.8000
```

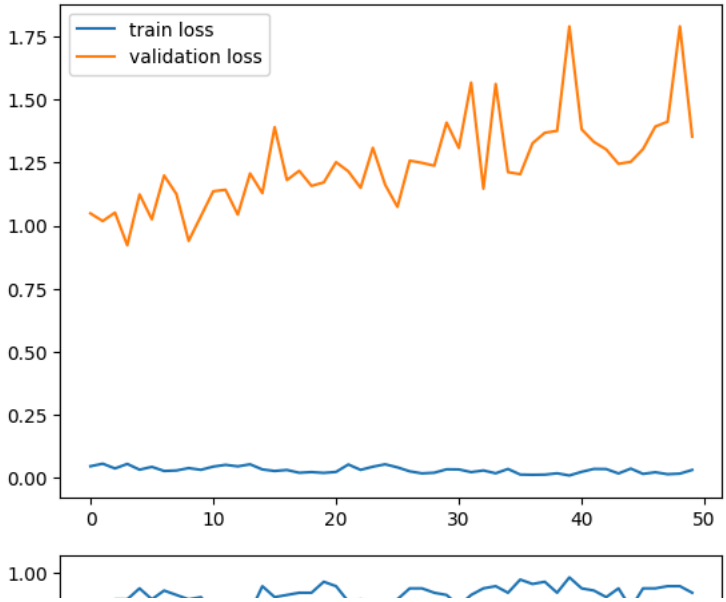
```
loss, accuracy = model.evaluate(X_test_scaled, Y_test)
print('Test Accuracy =', accuracy)
```

```
9/9 [==============================] - 2s 171ms/step - loss: 1.0779 - acc: 0.8517
Test Accuracy = 0.8517110347747803
```

```
h=history

# plot the loss value
plt.plot(h.history['loss'], label='train loss')
plt.plot(h.history['val_loss'], label='validation loss')
plt.legend()
plt.show()

# plot the accuracy value
plt.plot(h.history['acc'], label='train accuracy')
plt.plot(h.history['val_acc'], label='validation accuracy')
plt.legend()
plt.show()
```




```
input_image_path = input('Path of the image to be predicted: ')

input_image = cv2.imread(input_image_path)

cv2_imshow(input_image)

input_image_resized = cv2.resize(input_image, (128,128))

input_image_scaled = input_image_resized/255

input_image_reshaped = np.reshape(input_image_scaled, [1,128,128,3])

input_prediction = model.predict(input_image_reshaped)

print(input_prediction)


input_pred_label = np.argmax(input_prediction)

print(input_pred_label)


if input_pred_label == 1:

  print('The person has tumor')

elif input_pred_label == 2:
  print('The person has tumor')

elif input_pred_label == 3:
  print('The person has tumor')
```