Import Necessary Libraries

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import os
import tensorflow as tf
from PIL import Image

from sklearn.model_selection import train_test_split

from tensorflow.keras.utils import normalize, to_categorical
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Activation, Dropout, Flatten, Dense



from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
image_directory='/content/gdrive/MyDrive/brain_tumor_dataset/'
```

Load Data

```
image_directory='/content/gdrive/MyDrive/brain_tumor_dataset/'
no_tumor_images=os.listdir(image_directory+ 'no/')
yes_tumor_images=os.listdir(image_directory+ 'yes/')

print('No Tumor: ', len(no_tumor_images))
print('Tumor: ',len(yes_tumor_images))
```

```
No Tumor:  98
Tumor:  155
```

```
dataset=[]
label=[]

INPUT_SIZE=64
```

Create labels

```
for i , image_name in enumerate(no_tumor_images):
    if(image_name.split('.')[1]=='jpg'):
        image=cv2.imread(image_directory+'no/'+image_name)
        image=Image.fromarray(image,'RGB')
        image=image.resize((INPUT_SIZE,INPUT_SIZE))
        dataset.append(np.array(image))
        label.append(0)


for i , image_name in enumerate(yes_tumor_images):
    if(image_name.split('.')[1]=='jpg'):
        image=cv2.imread(image_directory+'yes/'+image_name)
        image=Image.fromarray(image, 'RGB')
        image=image.resize((INPUT_SIZE,INPUT_SIZE))
        dataset.append(np.array(image))
        label.append(1)


dataset=np.array(dataset)
label=np.array(label)

print('Dataset: ',len(dataset))
print('Label: ',len(label))
```

```
Dataset:  171
Label:  171
```

Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(dataset, label, test_size=0.2, random_state=42)
```

Normalize the Data

```
X_train = normalize(X_train, axis=1)
X_test = normalize(X_test, axis=1)
```

Model Building

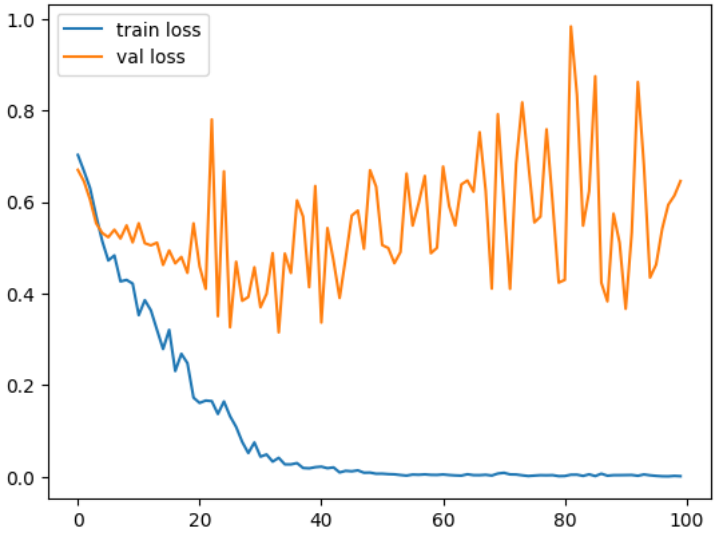
```
model=Sequential()

model.add(Conv2D(32, (3,3),activation='relu', input_shape=(INPUT_SIZE, INPUT_SIZE, 3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(32, (3,3),activation='relu', kernel_initializer='he_uniform'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64, (3,3),activation='relu', kernel_initializer='he_uniform'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1,activation='sigmoid'))


# compile the model
model.compile(loss='binary_crossentropy',optimizer='adam', metrics=['accuracy'])
```

Model Fit

```
r=model.fit(X_train, y_train,
batch_size=32,
verbose=1, epochs=100,
validation_data=(X_test, y_test),
shuffle=False)
```

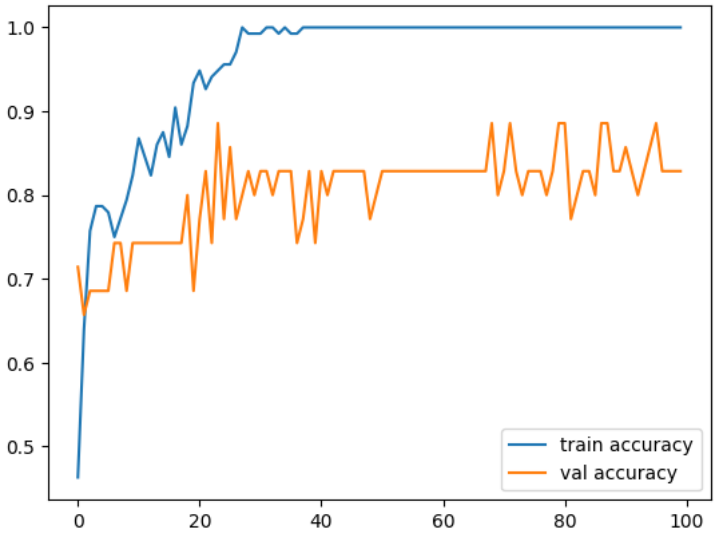
```
5/5 [==============================] - 0s 33ms/step - loss: 0.0036 - accuracy: 1.0000 - val_loss: 0.5335 - val_accuracy: 0.8286
Epoch 93/100
5/5 [==============================] - 0s 25ms/step - loss: 0.0019 - accuracy: 1.0000 - val_loss: 0.8618 - val_accuracy: 0.8000
Epoch 94/100
5/5 [==============================] - 0s 22ms/step - loss: 0.0048 - accuracy: 1.0000 - val_loss: 0.6792 - val_accuracy: 0.8286
Epoch 95/100
5/5 [==============================] - 0s 24ms/step - loss: 0.0029 - accuracy: 1.0000 - val_loss: 0.4347 - val_accuracy: 0.8571
Epoch 96/100
5/5 [==============================] - 0s 30ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.4631 - val_accuracy: 0.8857
Epoch 97/100
5/5 [==============================] - 0s 37ms/step - loss: 8.5535e-04 - accuracy: 1.0000 - val_loss: 0.5412 - val_accuracy: 0.82
Epoch 98/100
5/5 [==============================] - 0s 42ms/step - loss: 6.7611e-04 - accuracy: 1.0000 - val_loss: 0.5938 - val_accuracy: 0.82
Epoch 99/100
5/5 [==============================] - 0s 57ms/step - loss: 0.0016 - accuracy: 1.0000 - val_loss: 0.6138 - val_accuracy: 0.8286
Epoch 100/100
```

```
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')
```




```
<Figure size 640x480 with 0 Axes>
```

```
plt.plot(r.history['accuracy'], label='train accuracy')
plt.plot(r.history['val_accuracy'], label='val accuracy')
plt.legend()
plt.show()
plt.savefig('AccVal_accuracy')
```



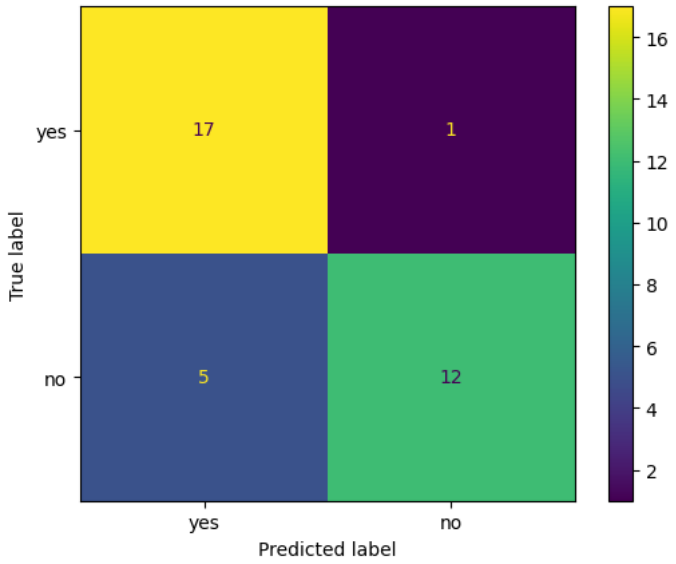

```
<Figure size 640x480 with 0 Axes>
```

```
from sklearn.metrics import accuracy_score, confusion_matrix, recall_score, f1_score
#from sklearn.metrics import plot_confusion_matrix


labels = ['yes,','no']
```

```
#confusion matrix
y_pred=model.predict(X_test)
y_true=y_test
```

```
2/2 [==============================] - 0s 5ms/step
```

```
import matplotlib.pyplot as plt
import numpy
from sklearn import metrics

actual = numpy.random.binomial(y_test,1)
predicted = numpy.random.binomial(1,y_pred)

confusion_matrix = metrics.confusion_matrix(actual, predicted)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = ['yes', 'no'])

cm_display.plot()
plt.show()
```



Save the Model

```
model.save('/content/gdrive/MyDrive/brain_tumor_dataset/BrainTumorDetection.h5')
```

Load Model

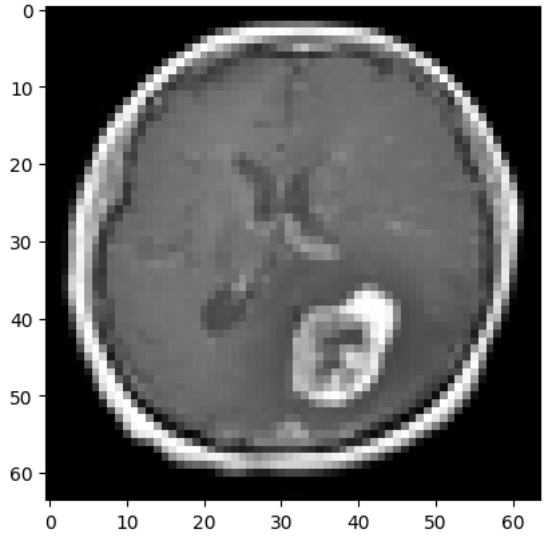
```
model = load_model('/content/gdrive/MyDrive/brain_tumor_dataset/BrainTumorDetection.h5')
```

Make Prediction on New Data

```
def make_prediction(img):

    input_img = np.expand_dims(img, axis=0)

    res = (model.predict(input_img) > 0.5).astype("int32")
    return res


def show_result(img):
    img_path = f"{image_directory}pred/{img}"
    image = cv2.imread(img_path)

    img = Image.fromarray(image)

    img = img.resize((64,64))

    img = np.array(img)

    plt.imshow(img)
```

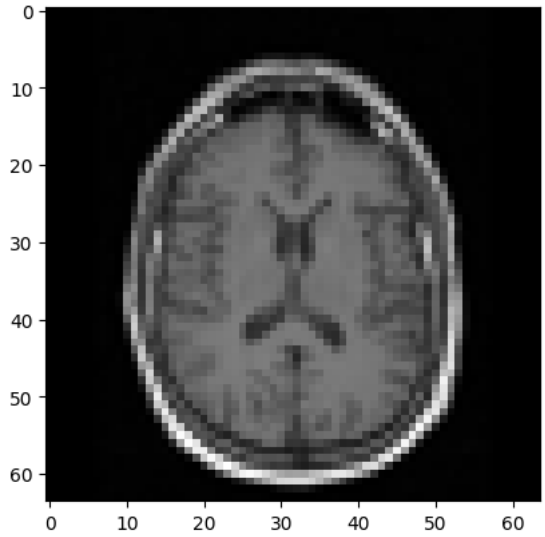
```
    plt.show()

    pred = make_prediction(img)
    if pred:
        print("yes Tumor Detected")
    else:
        print("No  Tumor Detected")
```

```
show_result('pred14.jpg')
```



```
1/1 [==============================] - 0s 170ms/step
yes Tumor Detected
```

```
show_result('pred1.jpg')
```



```
1/1 [==============================] - 0s 29ms/step
No  Tumor Detected
```

✓ 1m 19s completed at 5:33 PM