```python
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator

IMG_SIZE = 224
BATCH_SIZE = 32

TRAIN_datagen =
ImageDataGenerator(rescale=1./255,validation_split=0.2)

train_generator
=TRAIN_datagen.flow_from_directory('/content/drive/MyDrive/tumour',

target_size=(IMG_SIZE,IMG_SIZE),

batch_size=BATCH_SIZE,

class_mode='binary',
                                                subset='training')
```

Found 4221 images belonging to 1 classes.

```python
val_generator =
TRAIN_datagen.flow_from_directory('/content/drive/MyDrive/tumour',

target_size=(IMG_SIZE,IMG_SIZE),

batch_size=BATCH_SIZE,
                                            class_mode='binary',
                                            subset='validation')
```

Found 1055 images belonging to 1 classes.

```python
model=keras.Sequential([layers.Conv2D(32,
(3,3),activation='relu',input_shape=(IMG_SIZE,IMG_SIZE,3)),
                        layers.MaxPooling2D((2,2)),
                        layers.Conv2D(64,(3,3),activation='relu'),
                        layers.MaxPooling2D((2,2)),
                        layers.Conv2D(128,(3,3),activation='relu'),
                        layers.MaxPooling2D((2,2)),
                        layers.Flatten(),
                        layers.Dense(128,activation='relu'),
                        layers.Dense(1,activation='sigmoid')])
model.summary()
```

/usr/local/lib/python3.10/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.

```
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 109, 109, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_2 (Conv2D)	(None, 52, 52, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 128)	0
flatten (Flatten)	(None, 86528)	0
dense (Dense)	(None, 128)	11,075,712
dense_1 (Dense)	(None, 1)	129

 Total params: 11,169,089 (42.61 MB)

```
 Trainable params: 11,169,089 (42.61 MB)

 Non-trainable params: 0 (0.00 B)
```

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

```python
model.fit(train_generator, epochs=1, validation_data=val_generator, batch_size=BATCH_SIZE)
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/
data_adapters/py_dataset_adapter.py:122: UserWarning: Your `PyDataset`
class should call `super().__init__(**kwargs)` in its constructor.
`**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()
```

```
132/132 ──────────────────────── 735s 5s/step - accuracy: 0.9859 - loss:
0.0282 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
```

```
<keras.src.callbacks.history.History at 0x7b8073cbf940>
```

```python
model.save('/content/drive/MyDrive/tumour/braintumour.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.
```

```python
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
model=load_model('/content/drive/MyDrive/tumour/braintumour.h5')
print("model loaded")
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.
```

```
model loaded
```

```python
test_image_path='/content/drive/MyDrive/tumour/Brain_Tumor_Dataset/
Negative/Te-noTr_0003.jpg'
img=image.load_img(test_image_path,target_size=(224,224))
plt.imshow(img)
plt.axis()
plt.show()
```
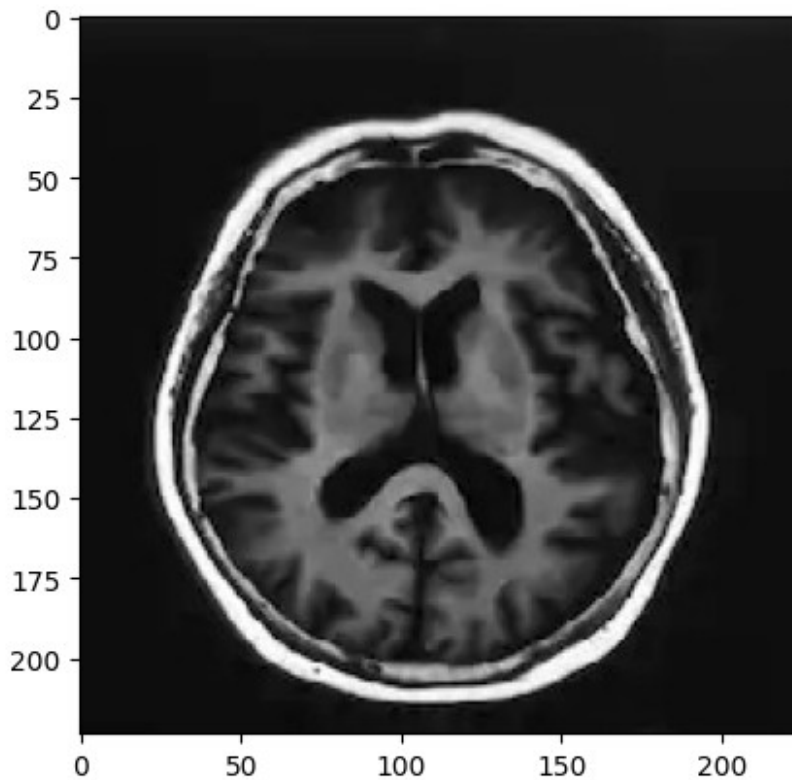
```
img_array=image.img_to_array(img)
img_array=np.expand_dims(img_array,axis=0)
img_array=img_array/255.

prediction=model.predict(img_array)
print(prediction)

1/1 ━━━━━━━━━━━━━━━━━ 0s 251ms/step
[[0.]]

if(prediction>0.5):
    print("tumour detected")
else:
  print("no tumour detected")

no tumour detected
```