

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

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

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force\_remount=True).

```
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/brain',
    target_size=(IMG_SIZE,IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training'
)
```

Found 800 images belonging to 1 classes.

```
val_generator=train_datagen.flow_from_directory(
    '/content/drive/MyDrive/brain',
    target_size=(IMG_SIZE,IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='validation'
)
```

Found 200 images belonging to 1 classes.

```
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')
])
```

```
/usr/local/lib/python3.11/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.summary()
```

```
Model: "sequential"
```

Layer (type) Param #	Output Shape
conv2d_3 (Conv2D) 896	(None, 222, 222, 32)
max_pooling2d_3 (MaxPooling2D) 0	(None, 111, 111, 32)
conv2d_4 (Conv2D) 18,496	(None, 109, 109, 64)
max_pooling2d_4 (MaxPooling2D) 0	(None, 54, 54, 64)
conv2d_5 (Conv2D) 73,856	(None, 52, 52, 128)
max_pooling2d_5 (MaxPooling2D) 0	(None, 26, 26, 128)
flatten (Flatten) 0	(None, 86528)
dense (Dense) 11,075,712	(None, 128)

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)

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

```
model.fit(train_generator,epochs=5,validation_data=val_generator,batch_size=BATCH_SIZE)
```

Epoch 1/5

25/25 ————— 115s 5s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val\_accuracy: 1.0000 - val\_loss: 0.0000e+00

Epoch 2/5

25/25 ————— 101s 4s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val\_accuracy: 1.0000 - val\_loss: 0.0000e+00

Epoch 3/5

25/25 ————— 150s 4s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val\_accuracy: 1.0000 - val\_loss: 0.0000e+00

Epoch 4/5

25/25 ————— 95s 4s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val\_accuracy: 1.0000 - val\_loss: 0.0000e+00

Epoch 5/5

25/25 ————— 94s 4s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val\_accuracy: 1.0000 - val\_loss: 0.0000e+00

<keras.src.callbacks.history.History at 0x7df77a9f2b10>

```
model.save('/content/drive/MyDrive/brain_model.h5')
```

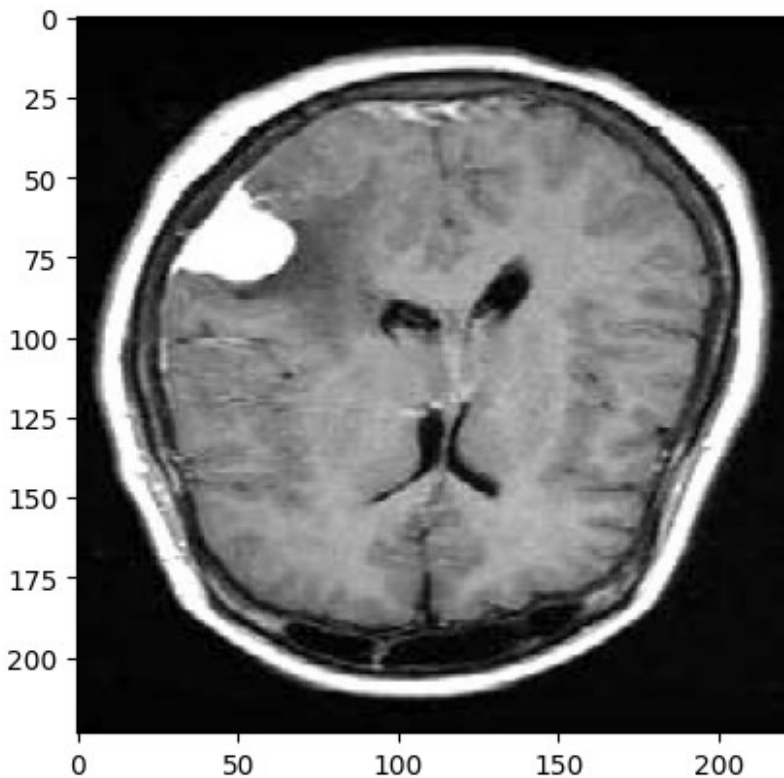
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save\_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my\_model.keras')` or `keras.saving.save\_model(model, 'my\_model.keras')`.

```
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/brain_model.h5')
print('Model Loaded Sucessfully')
```

```
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 Successfully

```
test_image_path="/content/drive/MyDrive/brain/train/yes/y155.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/=255

prediction=model.predict(img_array)
print(prediction)
if prediction>=0.5:
    print("Tumor Detected go and consult the doctor")
else:
    print("No Tumor Detected and no need to consult the doctor")
```

1/1 ————— 0s 154ms/step

[[0.]]

No Tumor Detected and no need to consult the doctor